This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

THIS PAGE BLANK (USPTO)

PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 :
G05B 13/02, 13/04

A1

(11) International Publication Number: WO 97/49011

(43) International Publication Date: 24 December 1997 (24.12.97)

(21) International Application Number: PCT/US97/10430

(22) International Filing Date: 13 June 1997 (13.06.97)

(30) Priority Data:
08/666,938 19 June 1996 (19.06.96) US

(71) Applicant: ARCH DEVELOPMENT CORPORATION [US/US]; 1101 East 58th Street, Chicago, IL 60637 (US).

(72) Inventors: GROSS, Kenneth, C.; 1433 Carriage Lane, Bolingbrook, IL 60440-1437 (US). WEGERICH, Stephan; 133 Golfview Drive, Glendale Heights, IL 60139 (US). SINGER, Ralph, M.; 1602 Brestol Court, Naperville, IL 60566-2781 (US). MOTT, Jack; 191 11th Street, Idaho Falls, ID 83401 (US).

(74) Agents: RECHTIN, Michael, D. et al.; Foley & Lardner, 777 E. Wisconsin Avenue, Milwaukee, WI 53202 (US).

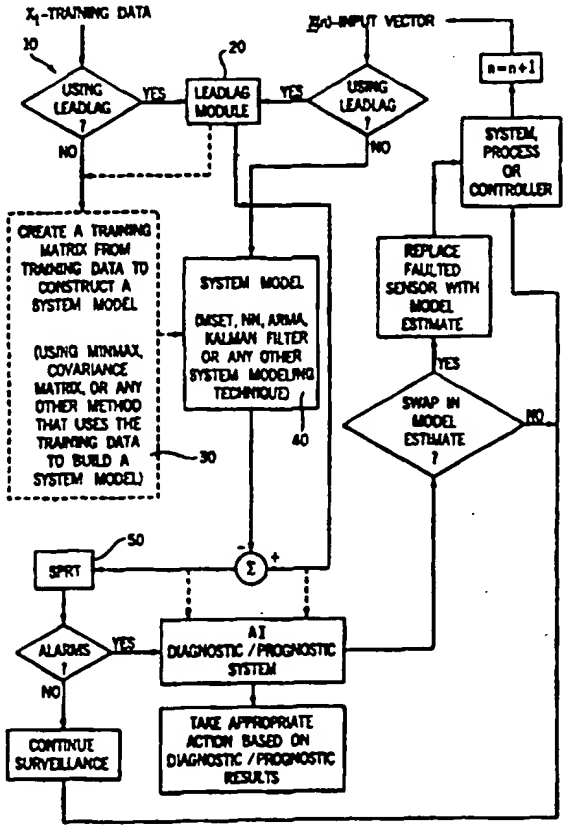
(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, HU, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZW, ARIPO patent (GH, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
With international search report.
With amended claims.

(54) Title: INDUSTRIAL PROCESS SURVEILLANCE SYSTEM

(57) Abstract

A system (10) and method for monitoring an industrial process and/or industrial data source (10). The system (10) includes a time correlation module (20), a training module (30), a system state estimation module (40) and a pattern recognition module (50). The system (10) generating time varying data sources, processing the data to obtain time correlation of the data (20), determining the range of data, determining learned states of normal operation (30) and using these states to generate expected values to identify a current state of the process closest to a learned, normal state (40); generating a set of modeled data, and processing the modeled data to identify a data pattern and generating an alarm (50) upon detecting a deviation from normalcy.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

Industrial Process Surveillance System

This invention was made with U.S. Government support under Contract No. W-31-109-ENG-38 awarded by the Department of Energy. The U.S. Government has certain rights in this invention.

The present invention is related generally to a method and system for carrying out surveillance of industrial processes using sensor or data source outputs. More particularly, the invention is concerned with a method and system for processing sensor data and using virtual data as an improved methodology over basic statistical approaches to industrial process surveillance. Further, the invention involves use of a plurality of techniques coupled for enhanced analysis of industrial process data.

Conventional parameter-surveillance schemes are sensitive only to gross changes in the mean value of a process or to large steps or spikes that exceed some threshold limit check. These conventional methods suffer from either large numbers of false alarms (if thresholds are set too close to normal operating levels) or a large number of missed (or delayed) alarms (if the thresholds are set too expansively). Moreover, most conventional methods cannot perceive the onset of a process disturbance, sensor deviation or data anomaly which gives rise to a signal below the threshold level for an alarm condition. Most methods also do not account for the relationship between a measurement by one sensor relative to another sensor measurement.

In another monitoring method, a conventional sequential probability ratio test ("SPRT") technique has found wide application as a signal validation tool in the nuclear reactor industry. The SPRT method is a pattern recognition technique which processes the stochastic components associated with physical process variables and has high sensitivity for the onset of subtle disturbances in those variables. Two features of the conventional SPRT technique make it attractive for parameter surveillance and fault detection: (1) early annunciation of the onset of a disturbance in noisy process variables, and (2) the SPRT technique has user-specificable false alarm and missed-alarm probabilities. SPRT techniques are primarily directed to the analysis of data from paired or multiple pairs of sensors in contrast to a large number of different process sensor data points. SPRT is also typically dependent on

assumptions of the data being independent of other data sources and being Gaussian distributed data. The SPRT technique used alone therefore has certain shortcomings in identifying anomalies in processes.

Other types of statistical techniques also have been developed for industrial process monitoring and analysis but have other insensitivities for certain classes of sensor data.

It is, therefore, an object of the invention to provide an improved method and system for surveillance of industrial processes and apparati.

It is another object of the invention to provide a novel method and system for on-line surveillance of industrial processes and apparati with multiple sensors.

It is also an object of the invention to provide an improved method and system for evaluation of process data, on-line or off-line, from sensors or data accumulation sources.

It is a further object of the invention to provide a novel method and system for performing preliminary analysis of data for alarm conditions prior to data input to a SPRT system.

It is an additional object of the invention to provide an improved method and system for masking selected sensor data and substituting virtual data to perform tests to determine whether abnormal process conditions or abnormal sensor conditions exist and whether or not to halt or modify the process under scrutiny.

It is still another object of the invention to provide a novel method and system using training data characteristic of normal system and/or sensor and/or data source operation to compare with ongoing industrial processes and/or data accumulation.

It is yet a further object of the invention to provide an improved method and system for processing data from a process to determine training data for normal operation, storing such training data on a computer storage media and analyzing real process data relative to the normal training data using a plurality of mathematical methodologies stored on a ROM or PROM storage medium.

It is also an additional object of the invention to provide a novel method and system utilizing a virtual signal characteristic of normal state operation derived on the basis of correlation with a plurality of other process data values to compare with a

real process data signal set for deriving the likelihood of an abnormal process or operation of data sources.

It is yet another object of the invention to provide a novel method and apparatus to accumulate training data to recognize any one of a plurality of specific states of operation and thereby identify a particular type of fault or condition present in a process or other system.

It is also a further object of the invention to provide a novel method and apparatus for monitoring a process using training data to identify slowly changing operational sensor data characteristic of normal process changes.

It is still an object of the invention to provide an improved method and system for determining whether a system or data source abnormality can be ignored without undesirable effects.

Other advantages and features of the invention, together with the organization and manner of operation thereof, will become apparent from the following detailed description when taken in conjunction with the accompanying drawings described below.

Brief Description of the Drawings

FIGURE 1 illustrates a schematic functional flow diagram of a preferred embodiment of the invention;

FIGURE 2 illustrates a functional flow diagram of a time lead-lag correlation methodology;

FIGURE 3 illustrates a functional flow diagram of a method of determining a full range of data by searching normal state training data;

FIGURE 4 illustrates a functional flow diagram of a method for modeling behavior of commercial system operating states;

FIGURE 5 illustrates a functional flow diagram of a method for performing pattern recognition;

FIGURE 6A illustrates sensor signals having a four second delay before applying a lead-lag method, and FIG. 6B illustrates the sensor signals after applying the lead-lag method;

FIGURE 7A illustrates sensor signal data from pump 1 power with an SMSET estimate superimposed thereon; FIG. 7B illustrates the SMSET estimation error

between the SMSET estimate and the sensor signal data; and FIG. 7C illustrates a histogram of the error;

FIGURE 8A illustrates sensor signal data from pump 2 power with an SMSET estimate superimposed thereon; FIG. 8B illustrates the SMSET estimation error between the SMSET estimate and the sensor signal data; and FIG. 8C illustrates a histogram of the error;

FIGURE 9A illustrates sensor signal data from pump 1 speed with an SMSET estimate superimposed thereon; FIG. 9B illustrates the SMSET estimation error between the SMSET estimate and the sensor signal data; and FIG. 9C illustrates a histogram of the error;

FIGURE 10A illustrates sensor signal data from pump 2 speed with an SMSET estimate superimposed thereon; FIG. 10B illustrates the SMSET estimation error between the SMSET estimate and the sensor signal data; and FIG. 10C illustrates a histogram of the error;

FIGURE 11A illustrates sensor signal data for reactor outlet flow rate; FIG. 11B illustrates the SMSET estimation error between the SMSET estimate and the sensor signal data; and FIG. 11C illustrates a histogram of the error;

FIGURE 12A illustrates sensor signal data for primary pump 2 flow rate; FIG. 12B illustrates the SMSET estimation error between the SMSET estimate and the sensor signal data; and FIG. 12C illustrates a histogram of the error;

FIGURE 13A illustrates sensor signal data for subassembly outlet temperature 1A1; FIG. 13B illustrates the SMSET estimation error between the SMSET estimate and the sensor signal data; and FIG. 13C illustrates a histogram of the error;

FIGURE 14A illustrates sensor signal data for subassembly outlet temperature 2B1; FIG. 14B illustrates the SMSET estimation error between the SMSET estimate and the sensor signal data; and FIG. 14C illustrates a histogram of the error;

FIGURE 15A illustrates sensor signal data for subassembly outlet temperature 4E1; FIG. 15B illustrates the SMSET estimation error between the SMSET estimate and the sensor signal data; and FIG. 15C illustrates a histogram of the error;

FIGURE 16A illustrates sensor signal data for subassembly outlet temperature 4F1; FIG. 16B illustrates the SMSET estimation error between the SMSET estimate and the sensor signal data; and FIG. 16C illustrates a histogram of the error;

FIGURE 17A illustrates sensor signal data for reactor outlet temperature 1534CF; FIG. 17B illustrates the SMSET estimation error between the SMSET estimate and the sensor signal data; and FIG. 17C illustrates a histogram of the error;

FIGURE 18A illustrates sensor signal data for primary tank sodium level 530 Float; FIG. 18B illustrates the SMSET estimation error between the SMSET estimate and the sensor signal data; and FIG. 18C illustrates a histogram of the error;

FIGURE 19A illustrates sensor signal data for primary tank sodium level 531 induction; FIG. 19B illustrates the SMSET estimation error between the SMSET estimate and the sensor signal data; and FIG. 19C illustrates a histogram of the error;

FIGURE 20A illustrates standard deviation of SMSET errors for each of the data in FIG. 7-19; and FIG. 20B illustrates the mean value of SMSET errors for each of the data in FIG. 7-19;

FIGURE 21 shows subassembly outlet temperature ("SOT") and SMSET estimates and in particular FIG. 21A illustrates time dependent normal SOT for 3F1 in the EBR-II nuclear reactor; FIG. 21B illustrates normal SOT for 3C1; FIG. 21C illustrates normal SOT for 5C2 and FIG. 21D illustrates normal SOT for 7A3;

FIGURE 22A-D illustrates SMSET estimation error for each of the data of FIGS. 21A-D, respectively;

FIGURE 23A-D illustrates SPRT results for each of the data of FIGS. 21A-D, respectively;

FIGURE 24A corresponds exactly to FIG. 21A; FIG. 24B includes a linear drift component compared to FIG. 21B; and FIGS. 24C and 24D correspond exactly to FIG. 21C and 21D, respectively;

FIGURE 25A corresponds exactly to FIG. 22A; FIG. 25B includes the effect on SMET estimation error of the linear drift of FIG. 24B; and FIGS. 24C and 24D correspond exactly to FIGS. 22C and 22D, respectively;

FIGURE 26A corresponds exactly to FIG. 23A; FIG. 26B illustrates the SPRT results for the linear drift error of FIG. 24B; and FIGS. 26C and D corresponds exactly to FIG. 23C and D, respectively;

FIGURES 27A and 28B corresponds exactly to FIGS. 21A and 21B, respectively; FIG. 27C includes a temporary amplitude pulse of 0.25% of the signal magnitude; and FIG. 27D corresponds exactly to FIG. 21D;

FIGURES 28A and 28B corresponds exactly to FIGS. 22A and 22B; FIG. 28C illustrates SMSET estimation error for the amplitude pulse effect of FIG. 27C and FIG. 27D corresponds exactly to FIG. 22D;

FIGURES 29A and 29B corresponds exactly to FIGS. 23A and 23B; FIG. 29C illustrates SPRT results of the amplitude pulse in FIG. 27C; and FIG. 29D corresponds exactly to FIG. 23D;

FIGURE 30A illustrates EBRII subassembly temperature data 3F1 but includes a uniform gain change compared to FIG. 21A and FIGS. 30B-D correspond exactly to FIGS. 21B-D;

FIGURE 31A illustrates the SMSET estimation error for the gain change of FIG. 30A; and FIGS. 31B-D correspond exactly to FIGS. 22B-D, respectively; and

FIGURE 32A illustrates the SPRT results for the gain change of FIG. 30A and SMSET analysis of FIG. 31A; and FIGS. 32B-D correspond exactly to FIGS. 23B-D, respectively.

Detailed Description of Preferred Embodiments

The system 10 herein includes a methodology (see FIG. 1) and apparatus for surveillance of sensor or data accumulation configurations in industrial, utility, business, medical, investment and transportation applications. The system 10 is useful for sensitive identification of the onset of sensor or data source degradation, process or system anomalies, or the onset of change to a different operational state. The most preferred form of the system 10 comprises a synergistic integration of four techniques to provide enhanced surveillance capabilities compared to conventional approaches (including neural networks), and also provide enhanced reliability and improved computational efficiency. The four elements that make up the most preferred surveillance form of the system 10 are embodied in four different methodologies generally characterized as a time correlation module 20, a training module 30, a system state estimation module 40 and a pattern recognition module 50.

Many attempts to apply conventional neural networks to nuclear, petrochemical or any industrial process surveillance applications have met with poor results in part because they fail to take into account lead-lag relationships (lack of proper time correlation of the data sets) between the various sensors or data sources.

In one example, a pipe is instrumented with a sequence of N thermocouples ("TCs") which measure the instantaneous temperature of the fluid passing through the pipe; and the signals from these TCs are displaced in time along the fluid stream flow path. If the sampling interval for the sensors is smaller than the transit time for fluid passing through the pipe, any attempt to monitor these signals with a neural net will produce very high uncertainties in the estimated signals for the TCs. For example, if it takes ten seconds for the fluid to pass through the pipe, and the sampling interval is one second, then at any given instant in time, TC(N), at the outlet end of the pipe, is seeing fluctuations that passed TC(1) ten seconds ago. These TCs may still contain a small degree of correlation due to gross changes in fluid temperature from a heat source or sink that is upstream of the pipe; however, the more valuable intersensor correlation that arises from local temperature perturbations carried along the pipe will be lost. This same phenomenon degrades the performance of neural networks and other pattern-recognition paradigms applied to any processes wherein the physical sensors or data sources are displaced in time across the process they are monitoring. Other examples of time delays in correlated systems include: systems with slow data flow rates and/or large physical distances (oil refineries, power plants, HVAC systems, and financial systems), delays due to analog or digital electronics (noise filters and large capacitors) or transmission delays (satellite communications, or transmitting data over different BUS systems.

In a preferred form of the time correlation model 20, a Leadlag component of the invention (see FIG. 6) performs dynamic, real-time intersensor lead-lag adjustments. The Leadlag module 20 performs adjustments so that the output signals, which are then input subsequently into the SMSET routine (the system state estimation module 40), are optimally correlated and impart the maximum information content to the pattern recognition module 50. The Leadlag module 20 is attached hereto as a computer software Appendix A. The Leadlag module 20 accomplishes the adjustment function by performing, for each pair of signals, an iterative regression procedure that generates a vector of correlation coefficients with respect to lag time. This vector of correlation coefficients is a unimodal concave function of lag time. Thus, the optimal lag time between the pair of signals is identified simply by

searching for the zero-crossing of the first derivative of the vector with respect to the lag time.

In other forms of the invention it may be unnecessary to utilize the Leadlag module 20, as noted in FIG. 1 wherein the option exists to skip use of the module 20. This could occur for systems in which there is adequate time correlation or if time shifting is not needed to achieve correlation. If the Leadlag module 20 is not utilized or the data has already been processed by the Leadlag module 20, the data is preferably input to a training module 30.

In a preferred embodiment this training module is a MiniMax module 30 which searches through all the observations for all signals or data during a training time period to construct training vectors that include the highest point and lowest point for each signal or data space under surveillance. A computer software Appendix B sets forth the MiniMax module 30. The MiniMax module 30 produces an "optimal" training set. It is optimal in the sense that it contains only, at most, 2N vectors, where N is the number of signals or data points in the system; and these vectors span the full range that all sensors or data sources have noted during the available training period. Wherever two or more sensors or data sources simultaneously attain maxima or minima, the resulting number of training vectors will be less than 2N.

In another form of the invention both the Leadlag module 20 and the MiniMax module 30 can be skipped, and the data can be input directly to the system state module 40.

Once the MiniMax module 30 has constructed a system model (or been skipped as noted above), the system state estimation module 40 (such as the preferred Subsystem Multivariate State Estimation Technique ("SMSET") module) models the behavior of a system through examples of the operating states of the commercial system being modeled. A computer sofflvare Appendix C sets forth the SMSET module 40. In general, the system state estimation module 40 can be any one of a variety of modeling methodologies, such as auto regressive moving average, a neural network, or a Kalman filtering technique or an empirical methodology.

The SMSET module 40 utilizes its memory of the learned states of the commercial system in conjunction with a single new observation to provide an

estimate of the current "true" system state. States of the system are represented by vectors whose elements are comprised of direct values of system parameters (measured signals) as well as any transformation of these system parameters that produce scalar values, e.g. calculated parameters based upon measured data. The SMSET module 40 does not require the state vector elements to be linearly independent as do most other types of estimation techniques. The learning process, which results in a "learned-state" matrix, is performed according to the MiniMax module 30 and the Leadlag module 20 described hereinbefore.

The basic methodology of the SMSET module 40 involves the input of a new observation of the behavior of a system that is compared with the "memory" of previous system behavior embodied in the learned-state matrix. A series of mathematical operations are performed that generates an estimate of the states in the system's memory that is "closest" to the new observation. The definition of "closest" that is used by the SMSET module 40 is the state that is lying closest to the new observation from the point of view of a set of rules that determine the association of two vectors. From this closest state, an estimate of the "true" state of the system is performed for each and every element of the state vector. Thus, given a set of current observed parameters of a system, the SMSET module 40 provides an estimate of the current true state of the system. The value of this method is that an estimate of all of the values of the system parameters in the state vector can be provided even if the current observation vector is incomplete (e.g. some sensors or data sources may have failed or are no longer available), contains erroneous or faulty elements (some sensors may have drifted, become uncalibrated, become contaminated with high noise levels, etc.), or even if the new system state does not coincide with previous operating states. However, the new system state must, in a general sense, be bounded by the domain of the states used to develop the system memory (learned-state matrix).

This estimation of the true current state of the commercial system, including estimated values of all system parameters, is used in conjunction with the actual measured system parameters to ascertain the operability of sensors (or other data sources) and disturbances in the system state. This state estimation process can further be described as an inference engine that accepts as input a set of learned states

and a new observation of the commercial system. After a series of operations are performed by the inference engine on this input, the result is an estimate of the learned state "closest" to the new observation. The definition of "closest" used here is the state lying closest to the new observation from the point of view of a set of rules that determine the association (overlap) of any two vectors. Another result is the estimation of the "true" value of each and every element in the new observation vector in the form of an estimated state vector. The series of operations performed in the inference engine consist of various matrix operations. First, all pairs of learned states are preferably associated two at a time using a rule set to create the elements of a recognition matrix. Next, the new observation is associated with each learned state using the rule set to produce a vector that has the same number of elements as the number of learned states. The largest element value in this vector identifies the "closest" learned state to the new observation. Finally, the normal matrix product of this vector with the recognition matrix produces a set of linear combination coefficients for combining the learned states into the estimated state vector. This methodology, when applied to any true state of a commercial system that is a combination of the learned states, yields a very close approximation to the true state. The actual closeness achieved depends most upon nonlinearities arising from the rule set and physical and/or random fluctuations in the variables and is demonstrated by direct testing. General experience with use of this method for real operating commercial systems has indicated predictive capabilities typically no worse than +/- 0.5% ad normally +/-0.1%.

Once the SMSET module 40 has modeled the data as described hereinbefore, the data is input to a pattern recognition module 50, such as the Sequential Probability Ratio Test ("SPRT") module. The computer software for the SPRT module 50 is in Appendix D. This SPRT module 50 is a sensitive pattern recognition method that can detect the onset of subtle degradation in noisy signals with high reliability, and with quantitative false-alarm and missed-alarm probabilities. Output from the SMSET module 40 is provided as a set of estimated signals (also called "virtual signals") for each sensor under surveillance. These virtual signals are fed into a network of interacting SPRT modules 50 together with the actual sensor readings. Each of the SPRT modules 50 receives one sensor-signal, virtual-signal pair. If any

sensor degradation or process disturbance starts to affect the output of one or more signals under surveillance, the SPRT module(s) 50 provide an annunciation to the operator and an actuator signal to the control system, which can selectively as needed automatically swap in the virtual signal to replace the degrading sensor signal, or data source. Further details of the SPRT module 50 are described in USPN 5,459,675, which is incorporated by reference herein.

The above-described combination of methodologies enables identification of a faulted process, a particular type of fault, a faulted sensor or data source or faulty data itself and enables actions to be taken to correct or modify the process being monitored.

In some cases when a failed sensor, or improper data stream source, is not important to the continued operation of a commercial system, the user can continue operating the commercial system or process if the sensor or data source were operating normally. For example, the system 10 can operate to substitute in a modeled estimate into a actual commercial system or process as input to replace a failed sensor or failed data source. This allows the commercial system or process to keep operating.

Since the system 10 does not rely on analytical modeling by itself, it is applicable to a wide variety of processes and systems, such as petro-chemical, power generation, automotive, manufacturing, medical, aeronautical, financial and any system in which signals are available for processing that are related to the commercial system process operation or performance. The only requirement of the system 10 is that there is some type of cross-correlation, be it linear or nonlinear, between the signals used as input to the system 10. The signals can be linear, nonlinear, stationary, nonstationary, clean or noisy (with an arbitrary distribution). The system 10 uses a database of historical operation data to model the commercial system or process. The database is assumed to contain data from all relevant operating modes of the system; however, if a new mode of operation is encountered and is determined not to be a result of commercial system or sensor failures, a new vector can be added to the existing training matrix to incorporate the unanticipated operating mode in the system model.

The following nonlimiting examples illustrate various aspects of the invention described herein. The data used is all taken from the EBR-II reactor at Argonne National Laboratory (West).

Example I

The effect of time delay correlation is described in this example, and FIGS. 6A and 6B show two voltage signals with a four second delay between them. Before the signals are processed by the Leadlag module 20 (see FIG. 6A), the correlation coefficient is 0.0182 which implies no correlation versus processing through the Leadlag module 20 to obtain a correlation of 0.9209 (see FIG. 6B). When the set of signals, or data, being used is more than two, all the possible pair combinations are used to calculate maximum possible correlation coefficients so all signals can be properly correlated.

Example II

An experiment to determine the accuracy of the invention (the "SMSET" methodology generally) was carried out using sensor data from the Experimental Breeder Reactor II (EBR-II) at Argonne National Laboratory (U558). The sensor data set contained 13 signals from sensors monitoring EBR-II. Table I shows the SMSET Estimation accuracy for EBR-II Data. Table I includes the channel numbers and descriptions for each of the sensor signals used in the analysis. The experiment was conducted in three steps; first the SMSET module was trained using two days worth of EBR-II data, next the trained SMSET module was used to estimate the state of approximately 110 hours worth of EBR-II data, and then the accuracy of the estimates was analyzed. For each of the sensor signals listed in Table I, FIGS 7-19, respectively, show the sensor signal (top plot) and SMSET estimate superimposed, the middle plot shows error between the SMSET and the sensor signal (in percent of the signal magnitude), ad a histogram (bottom plot) of the error. The histogram plots are compared to a Gaussian distribution with the same mean ad variance to give an

idea of how Gaussian the error signals. FIG. 20 provide a summary of the data of FIGS. 7-19.

A methodology entitled MiniMax (Appendix B) was used to train the system using the two days of training data cited above. After the MiniMax method was applied, a training matrix was constructed consisting of twenty-five unique vectors constituting an empirical model of the overall system. After creating the model, the methodology was then applied to the signals listed in the accuracy table. Each signal in the system has its own estimation error signal that is a measure of how close the pattern recognition model is representing the system
relative to the sensor readings. The second column of Table I lists the standard deviation of the estimate error for all of the signals in the experiment in terms of each of the signals' magnitude. The magnitude of the signal is defined by its mean during normal operation. The third column in Table I lists the mean of the estimate error for all of the signals also in terms of the signal magnitude. In general the estimate error standard deviations are in the range of 0.01% to 0.1% and the estimate error means are centered around 0. Bar graphs of the tabular information are shown in FIGS. 20A and 20B as graphic representation of the accuracy information.

Table I

SMSET Estimation Accuracy Table for EBRII Data:

Channel Number and Description	SMET Estimate Error Standard Diviation (% of Sensor Magnitude)	SMSET Estimate Error Mean Value (% of Sensor Magnitude)
1) Primary Pump #1 Power	0.05245	0.01241
2) Primary Pump #2 Power	0.14352	0.06595
3) Primary Pump #1 Speed	0.01078	0.00001
4) Primary Pump #2 Speed	0.01272	-0.00278
5) Reactor Outlet Flowrate	0.09585	0.04452
6) Primary Pump #2 Flowrate	0.06034	-0.02495
7) Subassembly Outlet Temperature 2B1	0.04635	0.00339

8) Subassembly Outlet Temperature 2B1	0.04904	-0.00118
9) Subassembly Outlet Temperature 4E1	0.05664	-0.00306
10) Subassembly Outlet Temperature 4F1	0.04926	-0.00413
11) Reactor Outlet Temperature 1534CF	0.04727	0.00513
12) Primary Tank Sodium	0.02440	-0.00280
12) Primary Tank Sodium Level 531 Induction	0.00615	0.00316

Example III

In FIGS. 21-32 examples of different sensor failure modes are shown along with how the system reacts to the failures. The preferred method of FIG. 1 is applied to the data. The sensor signals used in these examples are from a subset of 22 sensor signals used in the system. The 22 sensors monitored the EBR-II subassembly system at Argonne National Laboratory (West). Each of FIGS. 21-32 contains four subplots in which the upper most plot is related to Subassembly Outlet temperature ("SOT") 3F1, the upper middle plot is related to SOT 3C1, the lower middle plot is related to SOT 5C2, and the bottom plot is related to SOT 7A3. The system applied in each of the examples uses the same training matrix, which consists of 83 vectors selected from a training data base containing almost a weeks worth of data taken once every minute.

In FIGS. 21-23 are shown the results of using the system 10 during approximately 5.5 days of normal operation of EBR-II. FIG. 21 shows the SOT signals with their corresponding SMSET estimates (signal being the circles and the lines being the estimate). FIG. 22 shows the respective raw estimate errors (not in terms of the signal magnitude) derived by taking the difference between the SOR signals and corresponding SMSET estimates. Finally in FIG. 23 the results are shown from applying the decision making module of the system 10 (the SPRT module 50--

see Appendix D) to the SMSET estimation errors of FIG. 22. The SPRT plots show a total of only three false alarms which is a false alarm rate of 9.4×10^{-5}, and this is well within the specified false alarm rate of 1.0×10^{-3}.

One type of failure mode that is common among sensors is a slow drift. This type of failure can be difficult to detect early on especially when the sensor signals contain noise. FIGS. 24-26 illustrate a comparative example of processing data from this type of failure and failure identification. Signal #2 (FIG. 24B) has a 0.2% linear drift in it's mean over the 2.75 day period starting at 4000 minutes into the signal. The other sensors are operating normally. FIG. 25 shows the resulting SMSET estimation errors for each sensor signal. The error plot for signal #2 (FIG. 25B) shows evidence of drifting after the sensor signal has drifted approximately 0.05%. In FIG. 26 the SPRT method has determined that #2 (FIG. 26B) is drifting after approximately 0.05% of drift and that all other sensors are operating normally.

Another type of failure that can occur is a step change in the sensor signal. This can be the result of a short in the sensor or DAS, a calibration error or for a variety of other reasons. FIGS. 27-29, show a example of this type of failure for the SOT measurements. In this example sensor signal #3 (FIG. 27C) contains a pulse with an amplitude of 0.25% of the signal magnitude. The pulse starts at 4000 minutes and lasts for 2000 minutes. FIG. 27 shows the sensor signals and the SMSET estimates for the four SOT signals. FIG. 28 shows the resulting SMSET estimation errors. The error signal for #3 (FIG. 28C) shows that there is a problem starting at 4000 minutes and ending at 6000 minutes. The error signals are fed through the SPRT module 50, and the results are plotted in FIG. 29. Clearly, there has been a disturbance in sensor #3 (FIG. 29C) beginning at time 4000 minutes and ending at 6000 minutes.

In FIGS. 30-32 an example of a failure mode related to the sensor gain is shown. In this example the gain of the sensor signal changes over tune, i.e., the amplitude is increasing over time. The gain begins changing linearly over time from a beginning value of 1 to a final value of 1+0.075% of the sensor magnitude. The system 10 for the estimation error is applied to the signals, and the results are shown in FIG. 31. A human operator would most likely not be able to tell that there is a problem even after 8000 minutes by looking at the sensor signal. In FIG. 31A, it is

apparent that signal #1 is operating abnormally. This is confirmed in FIG. 32A by the SPRT results, showing a steadily increasing number of SPRT alarms over the 8000 minute period.

While preferred embodiments of the invention have been shown and described, it will be apparent to those skilled in the art that various changes and modifications can be made without departing from the invention in its broader aspects as set forth in the claims provided hereinafter.

Appendix A

Computer software for Leadlag module which performs dynamic, real-time intersensor lead-lag time correlation adjustments.

```
/*************************************************************/
/* LEADLAG OPTIMIZATION MODULE                               */
/*************************************************************/

#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>


#define pi 3.1415926
#define N 1024
#define xcsize 2047
#define fil_xcsize 2079
#define filsize 32

void phase_shift_optimization(int argc,char argv[][20],int method);
void conv(float f[], float g[], float c[], int size);
void der(float y[], float diff[], int size);
void roots(float diffxc[], int size, float *root);
void fliplr(float a[], int size);
void prn_info(char *);


void main(int argc, char **argv)
{
 int i,method;
 char tempargv[10][20];

 if(argc !=5) {
   prn_info(argv[0]);
   exit(1);
  }

 for(i=0;i<argc;i++)
  strcpy(tempargv[i],argv[i]);

 printf("Enter 1 or 2 below.  (\"1\" means the employment of the");
 printf(" derivative technique to find the shift, while \"2\"");
 printf(" means the application of direct max. correlation");
 printf(" technique.):\n");
 scanf("%d",&method);

 phase_shift_optimization(argc,tempargv,method);

}



void phase_shift_optimization(int argc, char argv[][20], int method)
{
 int i,j,ph;
 float root;
 float f1[xcsize],f2[xcsize],b[fil_xcsize];
 float xc[fil_xcsize], fil_xc[fil_xcsize];
 float diff[fil_xc[fil_xcsize-1];
 FILE *infile1,*infile2,*infile3,*outfile, *outfile1, *outfile2;
```

INTERNATIONAL SEARCH REPORT

International application No
.PCT/US97/10430

A. CLASSIFICATION OF SUBJECT MATTER
IPC(6) :G05B 13/02, 13/04
US CL :364/148, 149, 164
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED
Minimum documentation searched (classification system followed by classification symbols)
U.S . 364/148, 149, 164

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
APS

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No
Y,P	US 5,586,066 A (WHITE ET AL) 17 December 1996, columns 1-3, 5 and 7.	1-16
Y	US 5,459,675 A (GROSS ET AL) 17 October 1995, figures 1-8A.	1-16
Y	US 5,410,492 A (GROSS ET AL) 25 April 1995, figures 1-8A and columns 3-7.	1-16
Y	US 5,402,521 A (NIIDA ET AL) 28 March 1995, columns 1-3.	1 and 10

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance, the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance, the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search	Date of mailing of the international search report
03 AUGUST 1997	0 3 SEP 1997
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C 20231 Facsimile No. (703) 305-3230	Authorized officer REBA I. ELMORE Telephone No (703) 305-9706

Form PCT/ISA/210 (second sheet)(July 1992)*

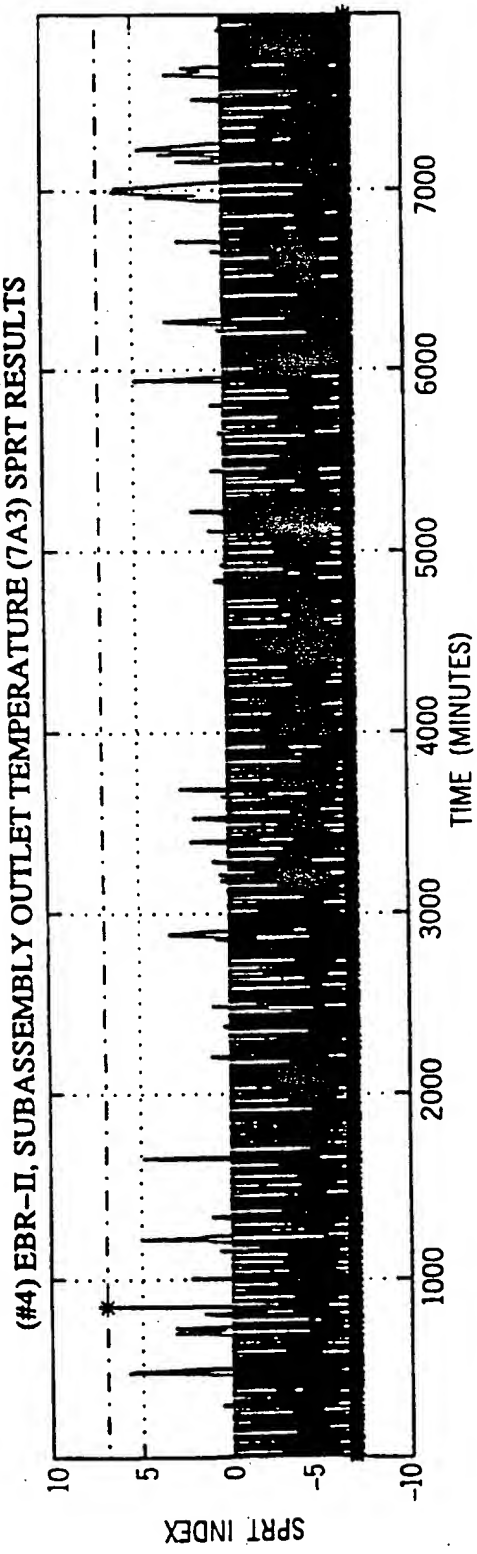

(#4) EBR-II, SUBASSEMBLY OUTLET TEMPERATURE (7A3) SPRT RESULTS
SPRT INDEX
10
5
0
-5
-10
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)

FIG. 32D

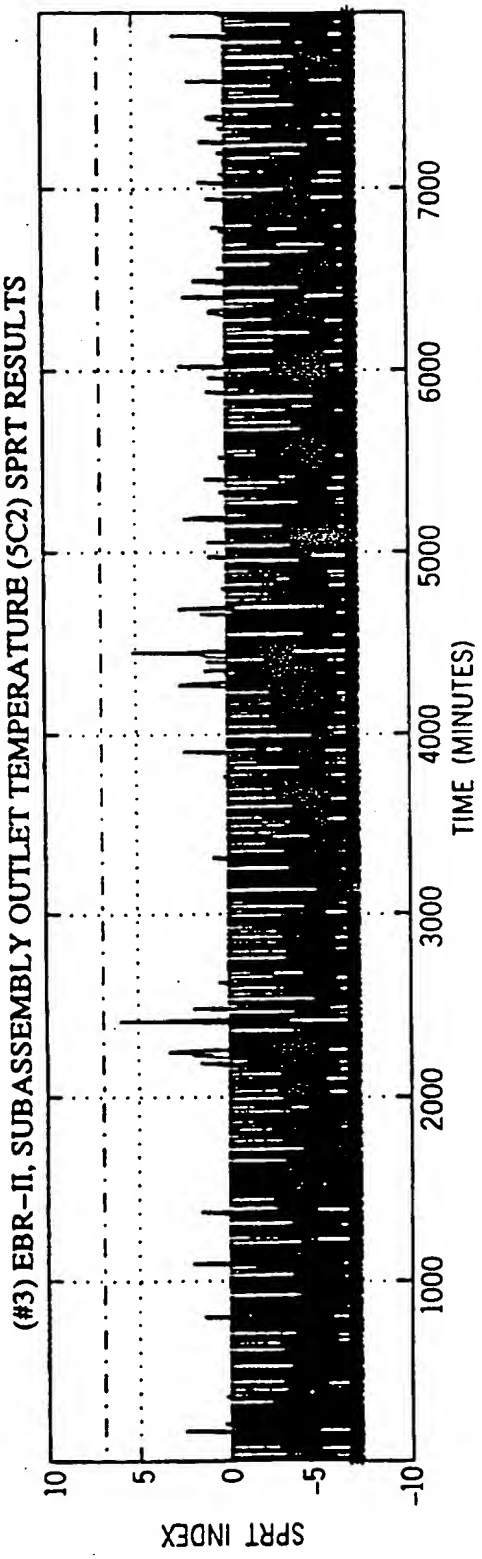

(#3) EBR-II, SUBASSEMBLY OUTLET TEMPERATURE (5C2) SPRT RESULTS
SPRT INDEX
10
5
0
-5
-10
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)

FIG. 32C

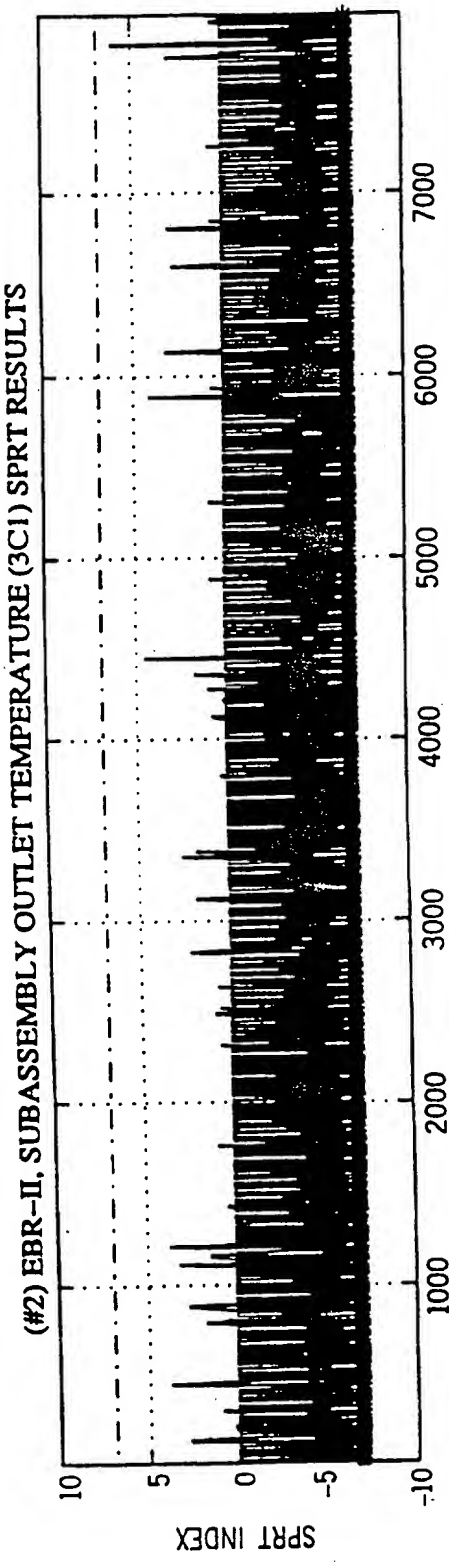

(#2) EBR-II, SUBASSEMBLY OUTLET TEMPERATURE (3C1) SPRT RESULTS
SPRT INDEX
10
5
0
-5
-10
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)

FIG. 32B

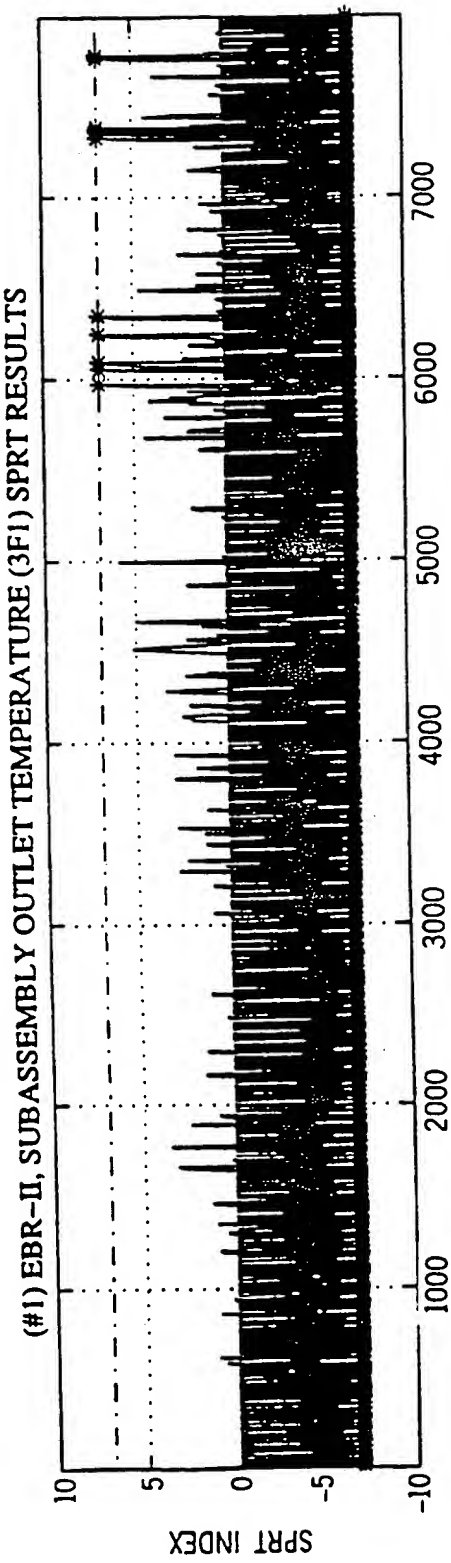

(#1) EBR-II, SUBASSEMBLY OUTLET TEMPERATURE (3F1) SPRT RESULTS
SPRT INDEX
10
5
0
-5
-10
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)

FIG. 32A

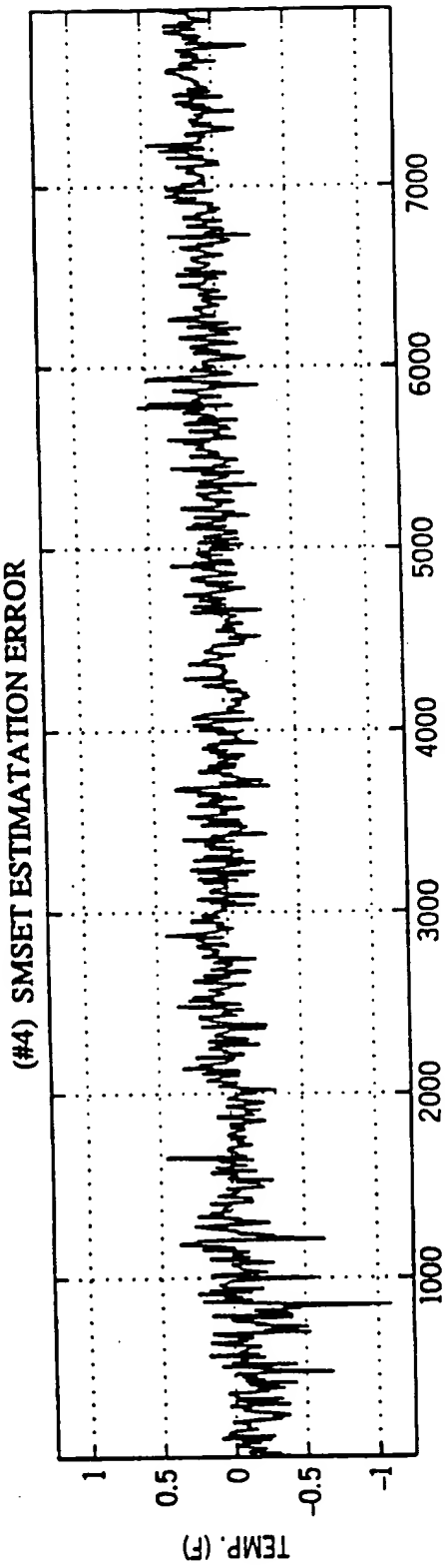

(#4) SMSET ESTIMATATION ERROR
TEMP. (F)
1
0.5
0
-0.5
-1
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)

FIG. 31D

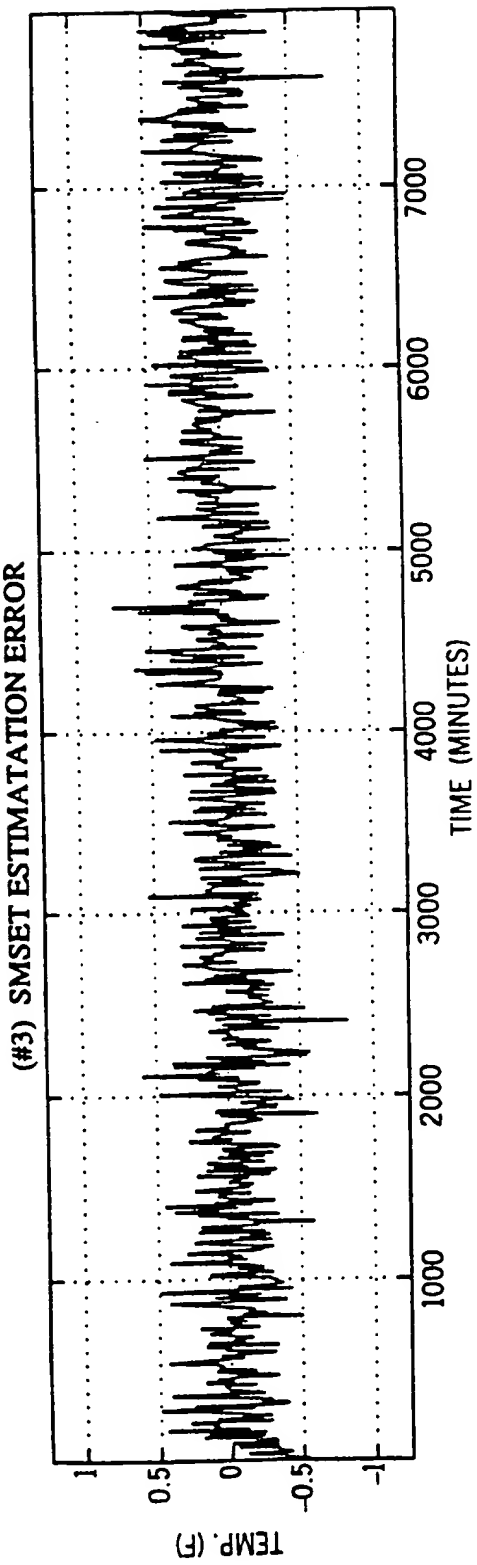

(#3) SMSET ESTIMATATION ERROR
TEMP. (F)
1
0.5
0
-0.5
-1
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)

FIG. 31C

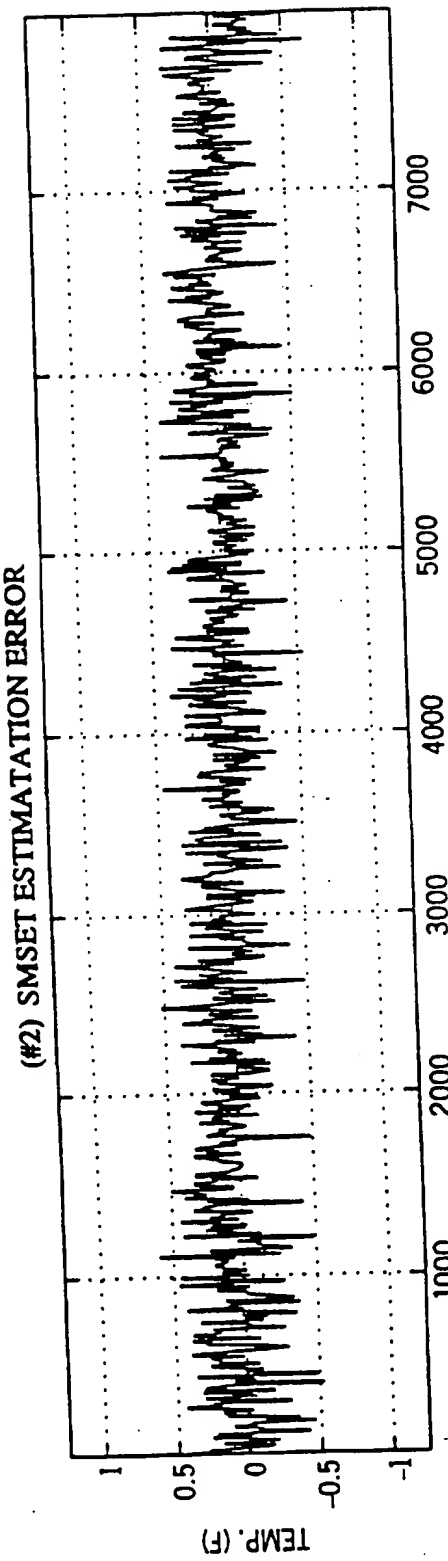

(#2) SMSET ESTIMATATION ERROR
TEMP. (F)
1
0.5
0
-0.5
-1
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)

FIG. 31B

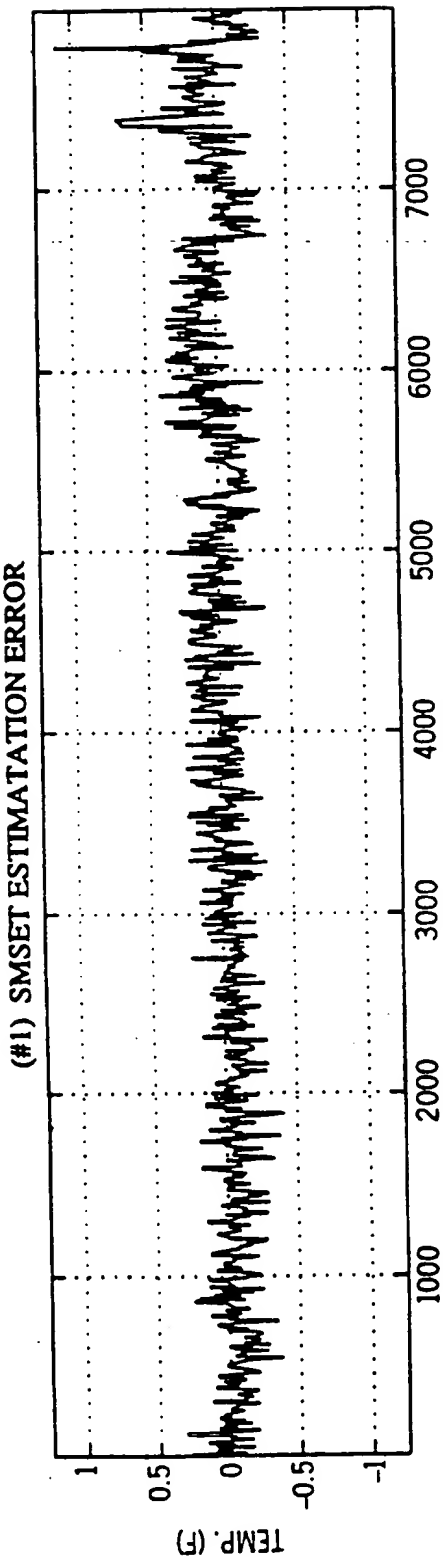

(#1) SMSET ESTIMATATION ERROR
TEMP. (F)
1
0.5
0
-0.5
-1
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)

FIG. 31A

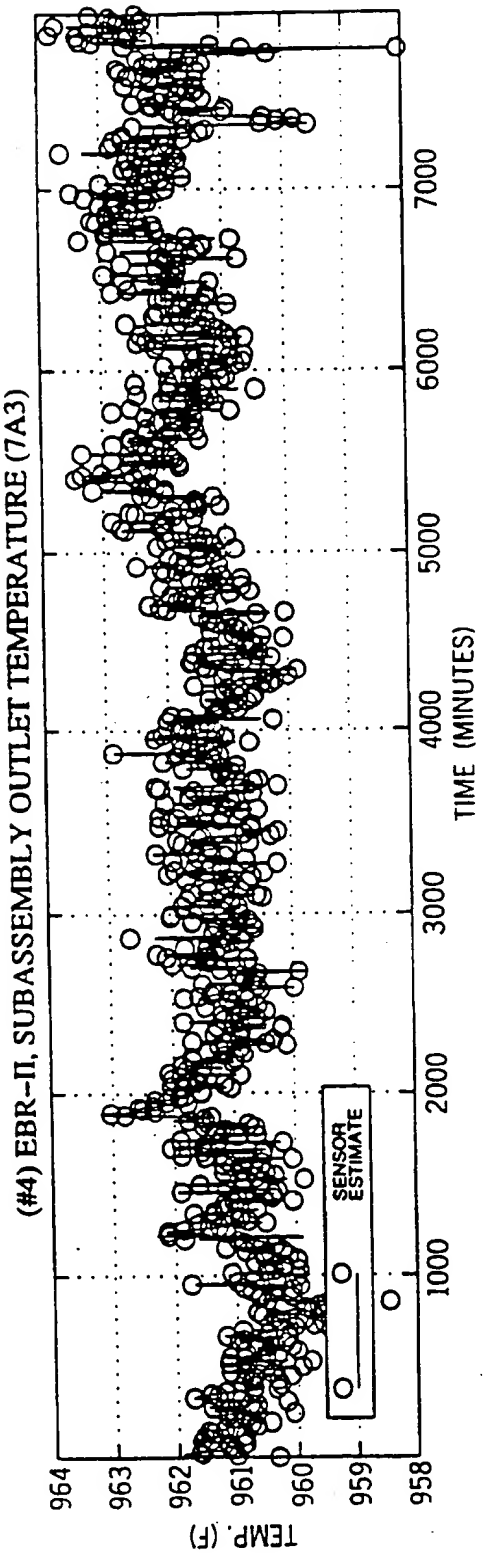

(#4) EBR-II, SUBASSEMBLY OUTLET TEMPERATURE (7A3)
TEMP. (F)
964
963
962
961
960
959
958
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)
SENSOR
ESTIMATE

FIG. 30D

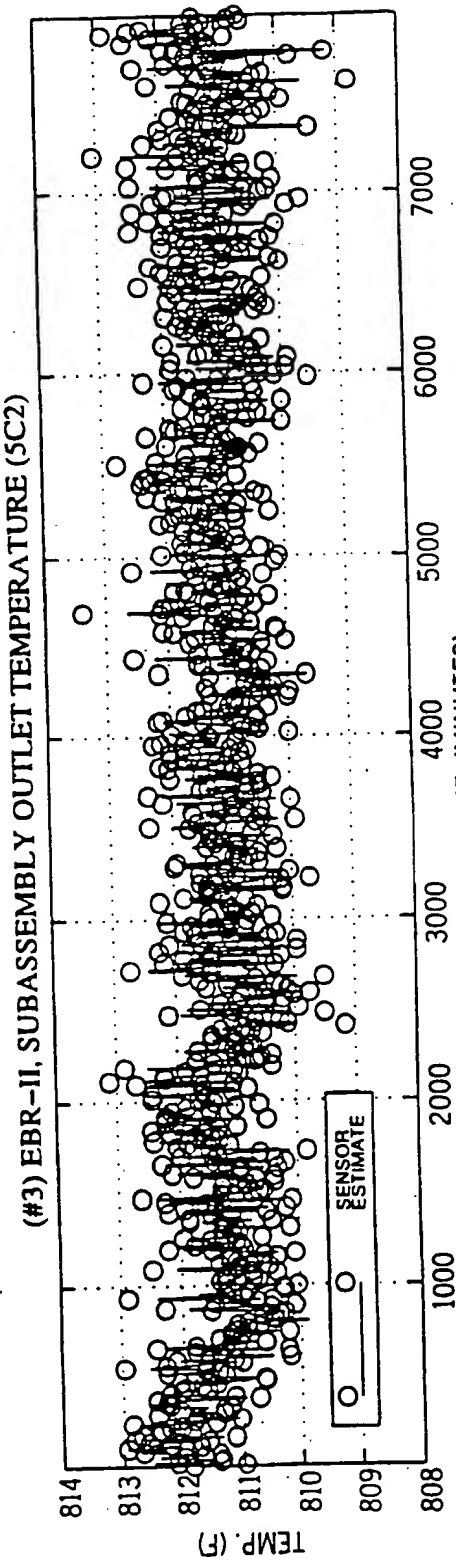

(#3) EBR-II, SUBASSEMBLY OUTLET TEMPERATURE (5C2)
TEMP. (F)
814
813
812
811
810
809
808
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)
SENSOR
ESTIMATE

FIG. 30C

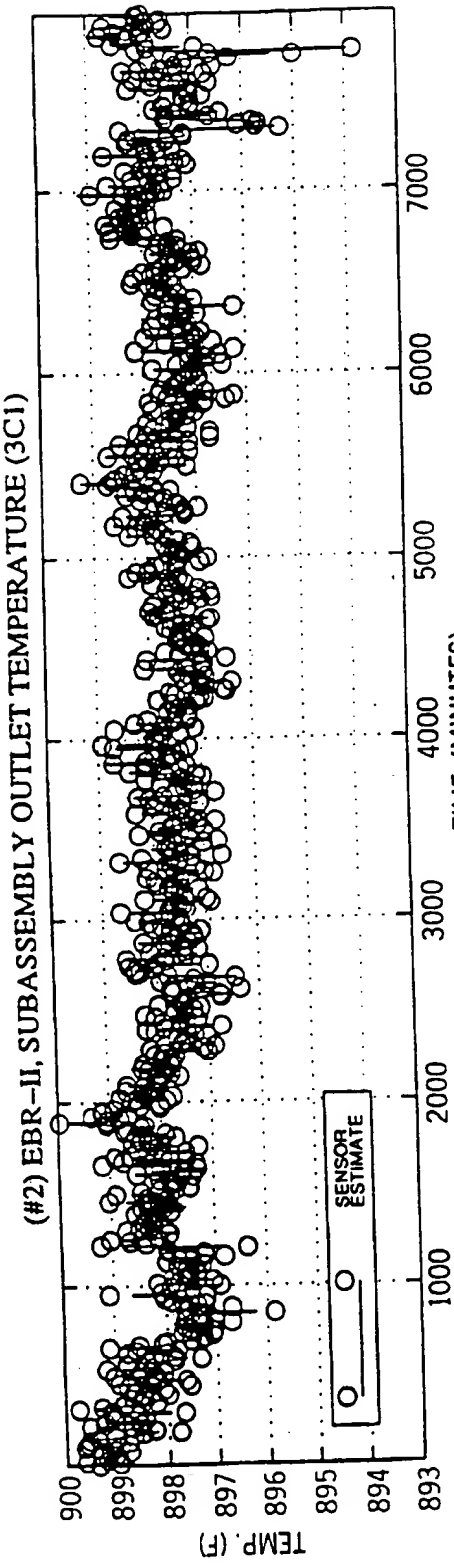

(#2) EBR-II, SUBASSEMBLY OUTLET TEMPERATURE (3C1)
TEMP. (F)
900
899
898
897
896
895
894
893
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)
SENSOR
ESTIMATE

FIG. 30B

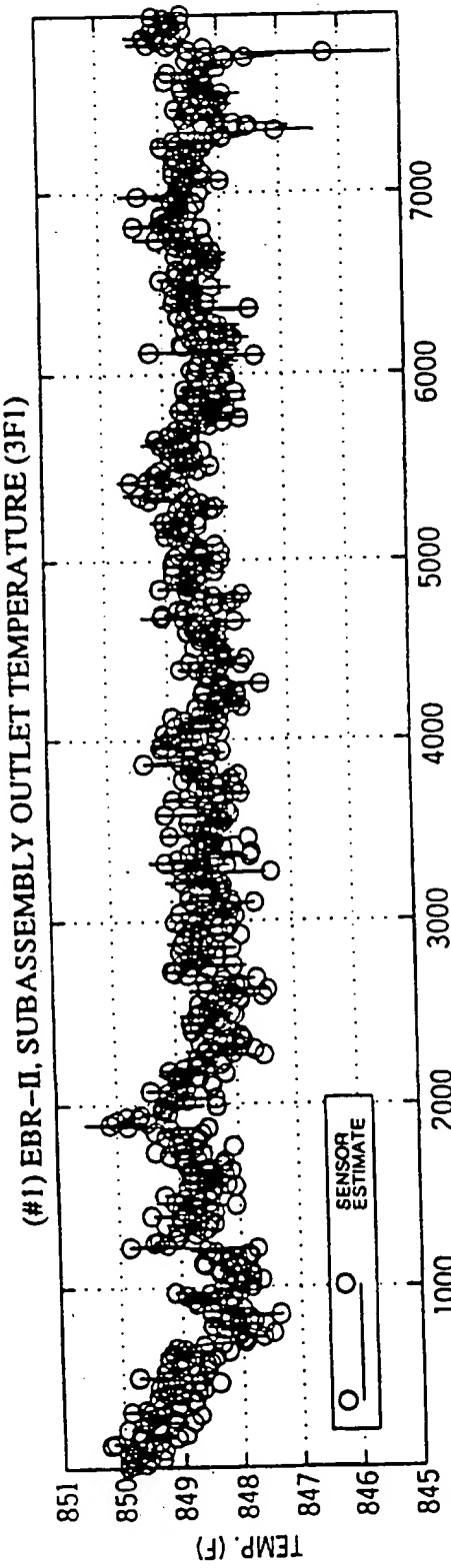

(#1) EBR-II, SUBASSEMBLY OUTLET TEMPERATURE (3F1)
TEMP. (F)
851
850
849
848
847
846
845
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)
SENSOR
ESTIMATE

FIG. 30A

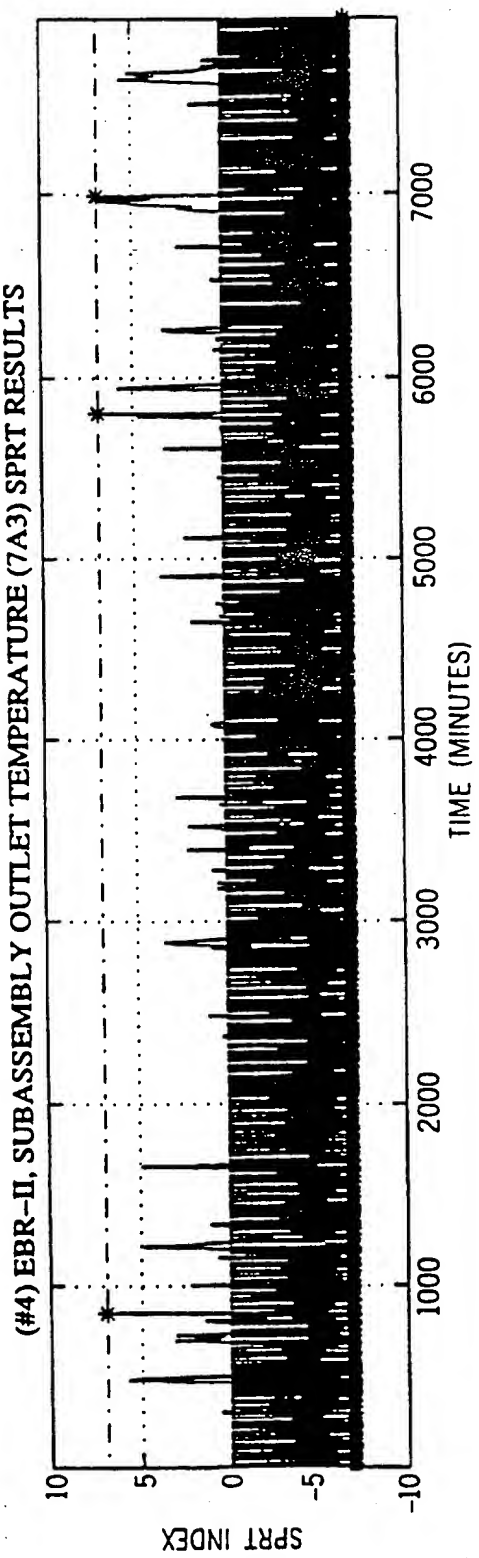

(#4) EBR-II, SUBASSEMBLY OUTLET TEMPERATURE (7A3) SPRT RESULTS
SPRT INDEX
10
5
0
-5
-10
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)

FIG. 29D

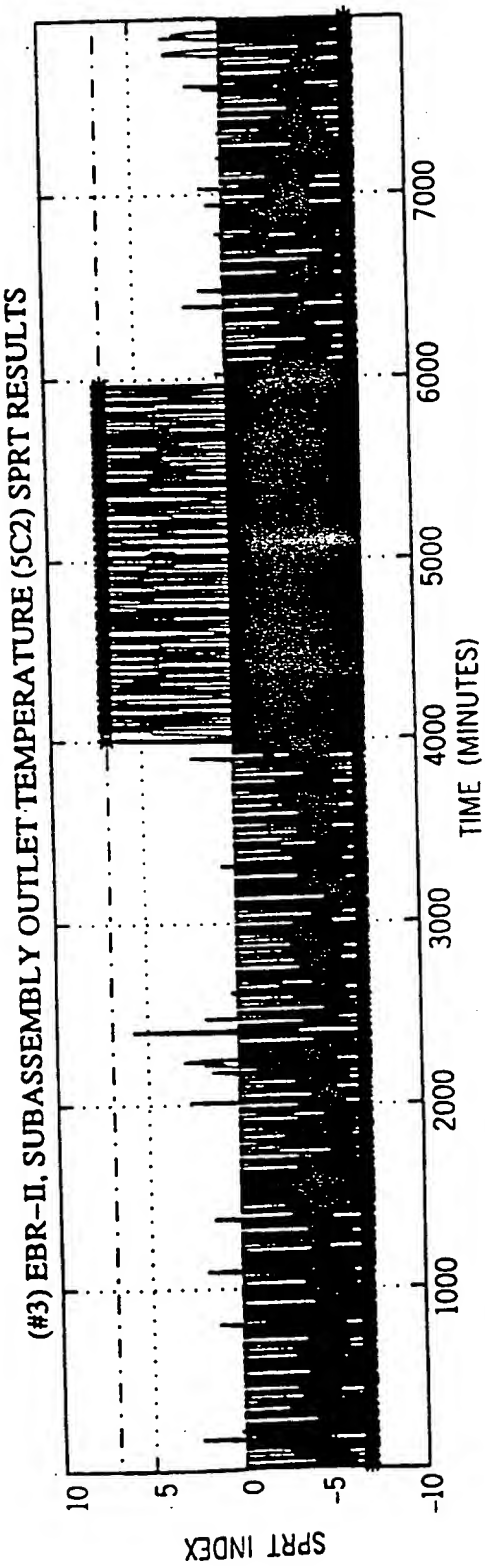

(#3) EBR-II, SUBASSEMBLY OUTLET TEMPERATURE (5C2) SPRT RESULTS
SPRT INDEX
10
5
0
-5
-10
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)

FIG. 29C

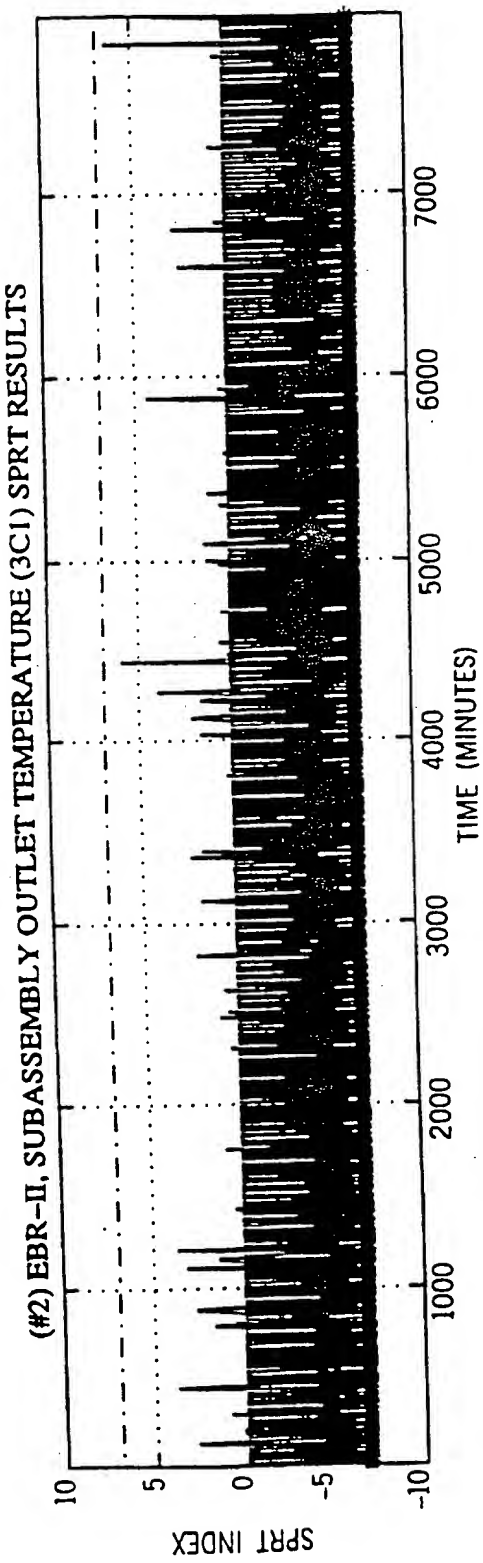

(#2) EBR-II, SUBASSEMBLY OUTLET TEMPERATURE (3C1) SPRT RESULTS
SPRT INDEX
10
5
0
-5
-10
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)

FIG. 29B

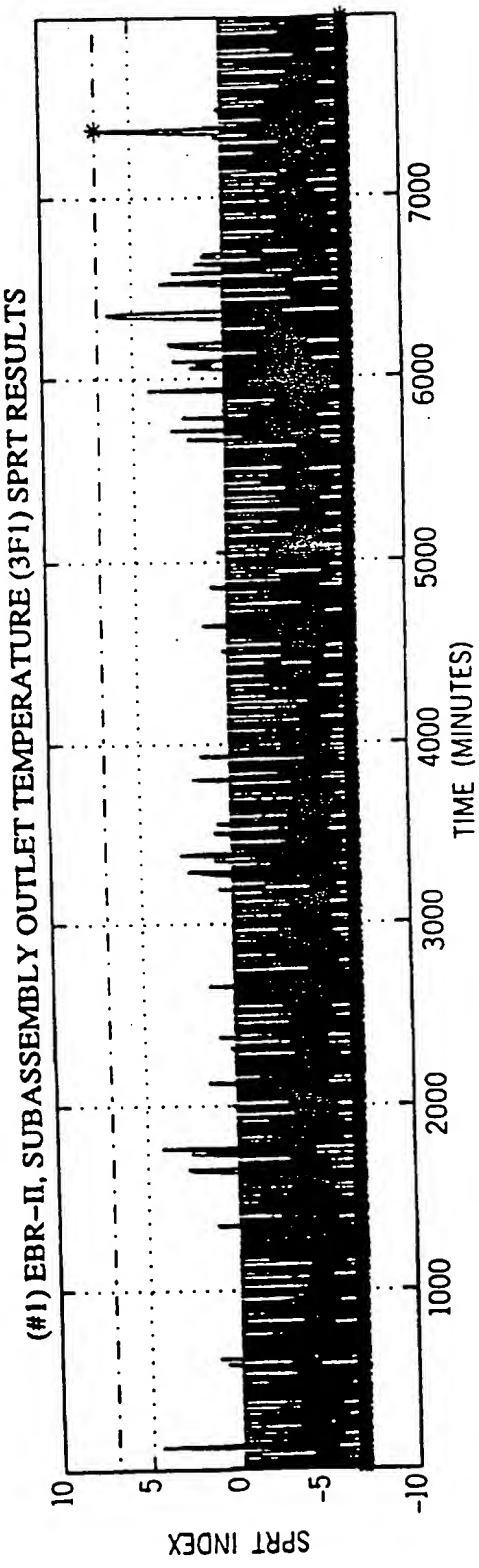

(#1) EBR-II, SUBASSEMBLY OUTLET TEMPERATURE (3F1) SPRT RESULTS
SPRT INDEX
10
5
0
-5
-10
TIME (MINUTES)
1000
2000
3000
4000
5000
6000
7000

FIG. 29A

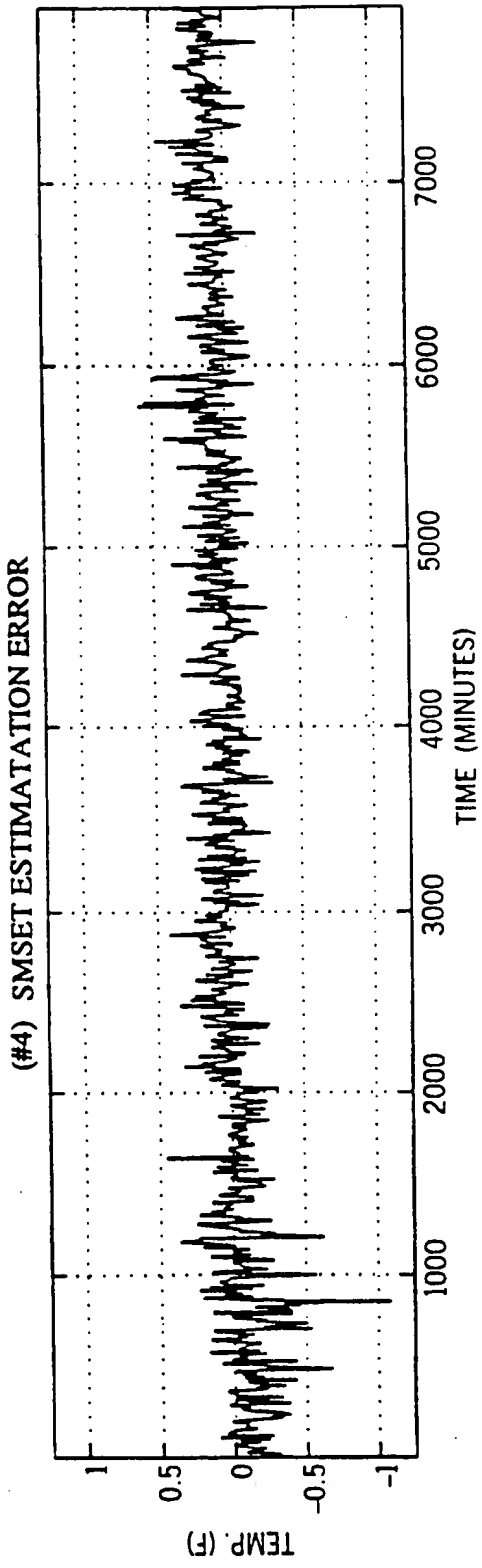

(#4) SMSET ESTIMATATION ERROR
TEMP. (F)
1
0.5
0
-0.5
-1
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)

FIG. 28D

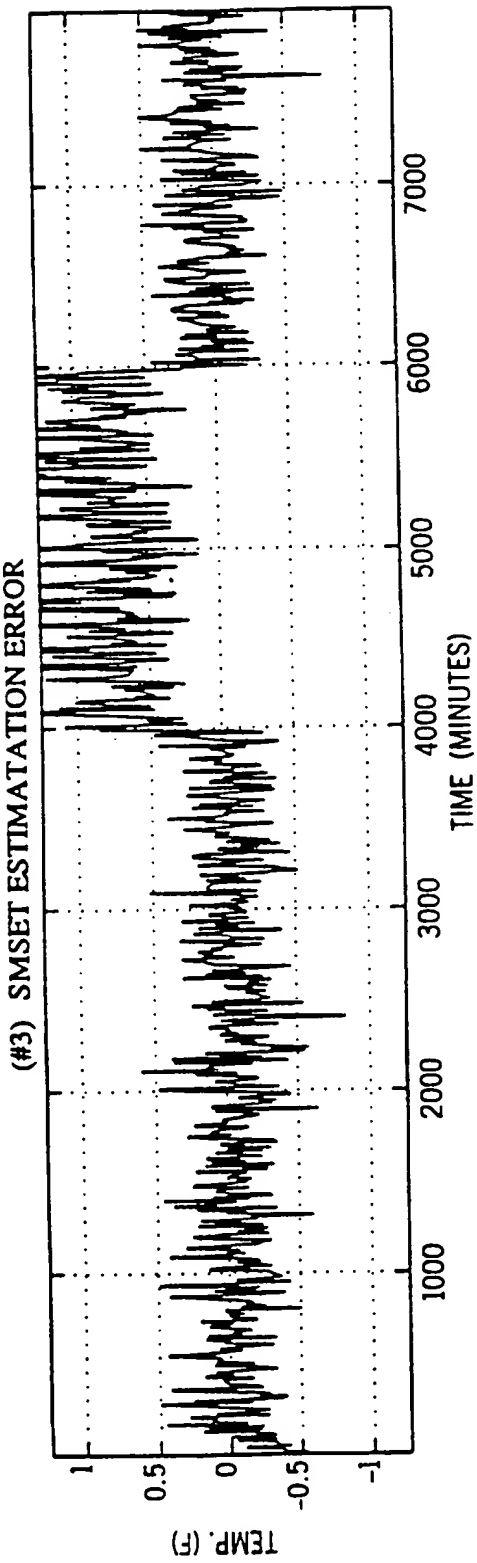

(#3) SMSET ESTIMATATION ERROR
TIME (MINUTES)
TEMP. (F)
1000
2000
3000
4000
5000
6000
7000
1
0.5
0
-0.5
-1

FIG. 28C

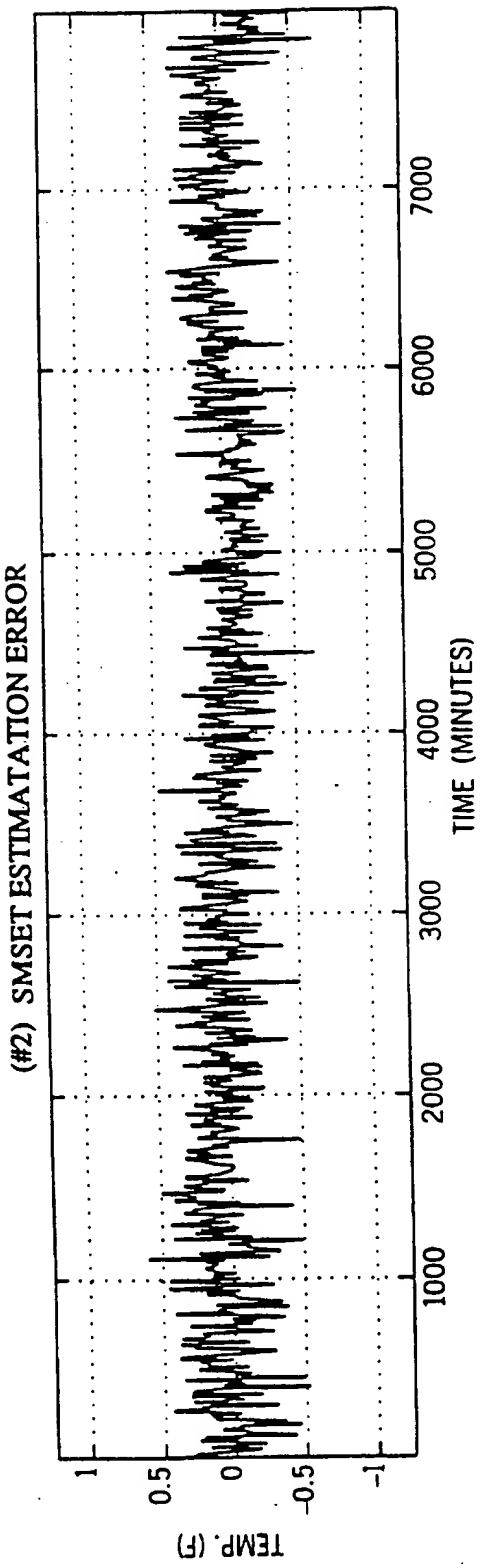

(#2) SMSET ESTIMATATION ERROR
TEMP. (F)
1
0.5
0
-0.5
-1
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)

FIG. 28B

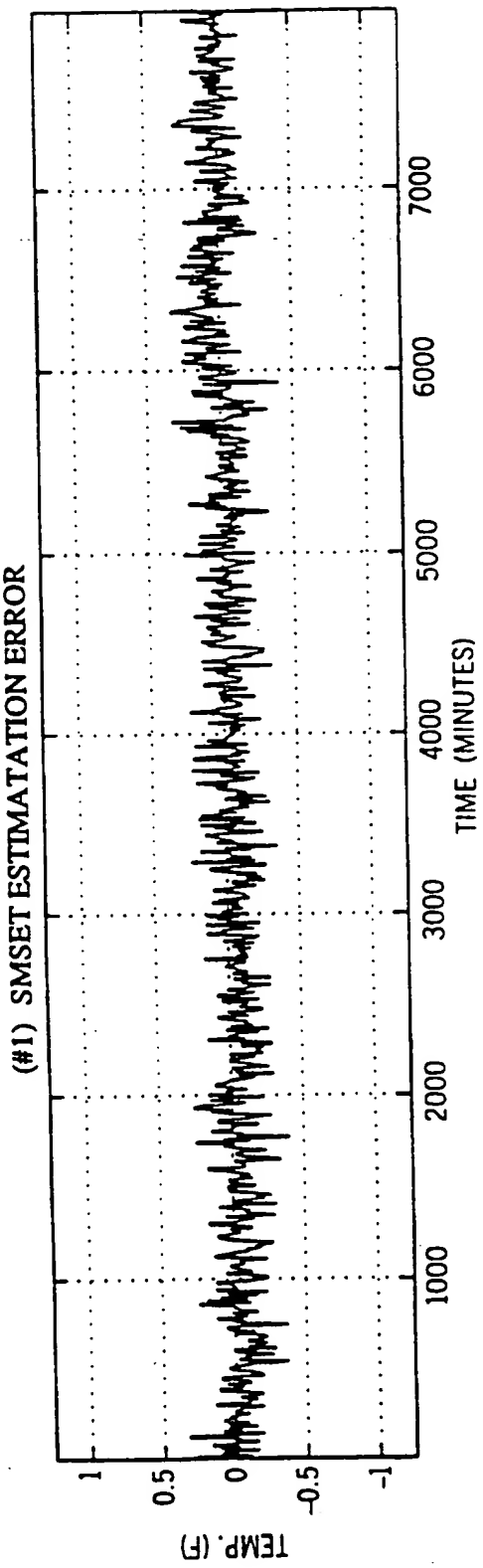

(#1) SMSET ESTIMATATION ERROR
TEMP. (F)
1
0.5
0
-0.5
-1
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)

FIG. 28A

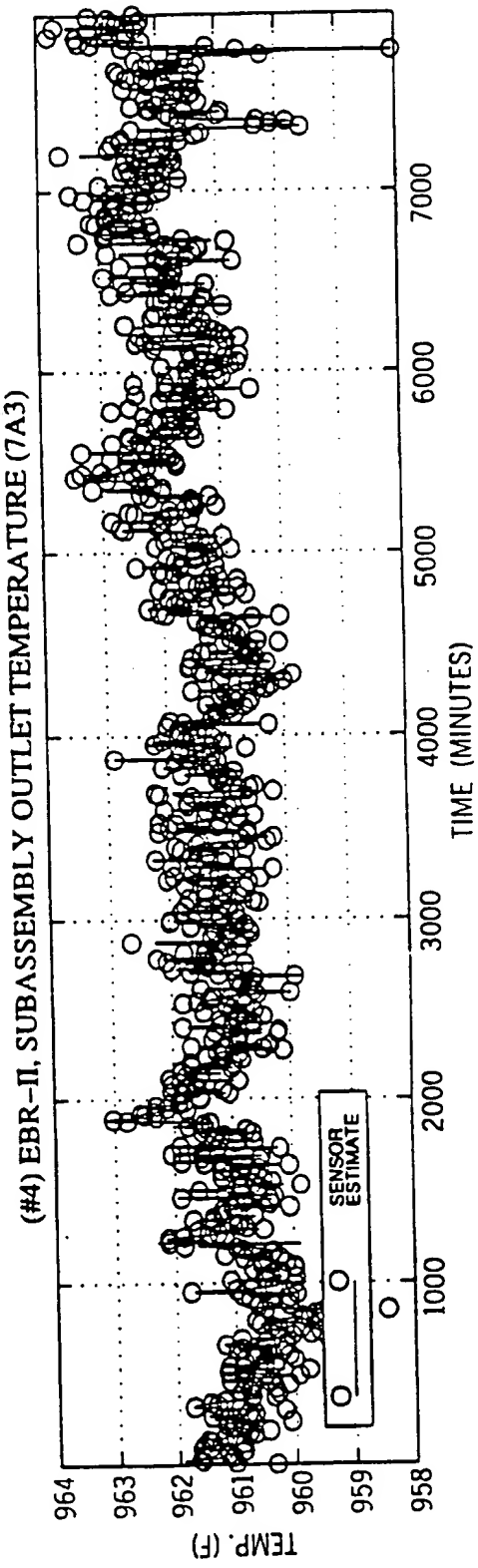

(#4) EBR-II, SUBASSEMBLY OUTLET TEMPERATURE (7A3)
TEMP. (F)
964
963
962
961
960
959
958
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)
SENSOR
ESTIMATE

FIG. 27D

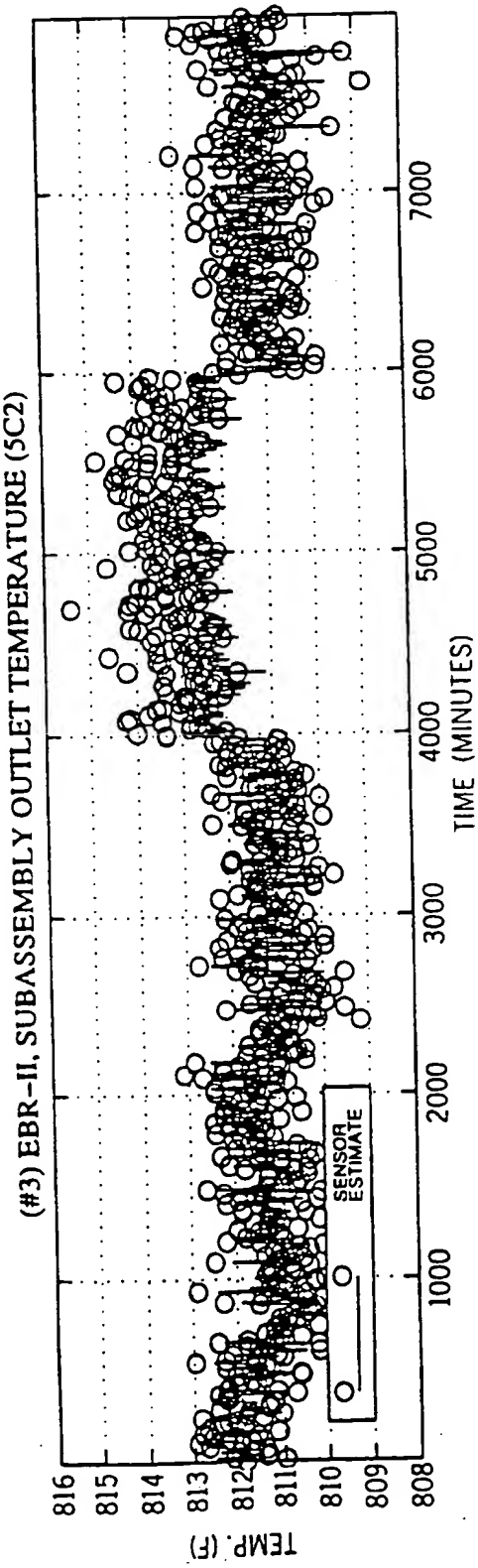

(#3) EBR-II, SUBASSEMBLY OUTLET TEMPERATURE (5C2)
TEMP. (F)
816
815
814
813
812
811
810
809
808
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)
SENSOR
ESTIMATE

FIG. 27C

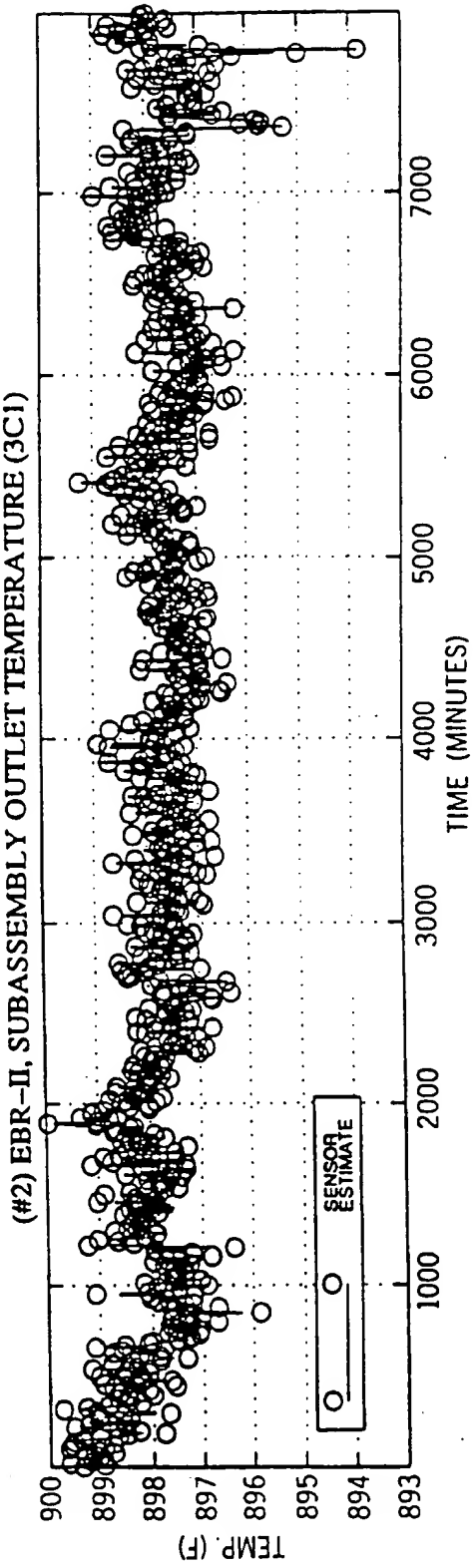

(#2) EBR-II, SUBASSEMBLY OUTLET TEMPERATURE (3C1)
TEMP. (F)
900
899
898
897
896
895
894
893
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)
SENSOR
ESTIMATE

FIG. 27B

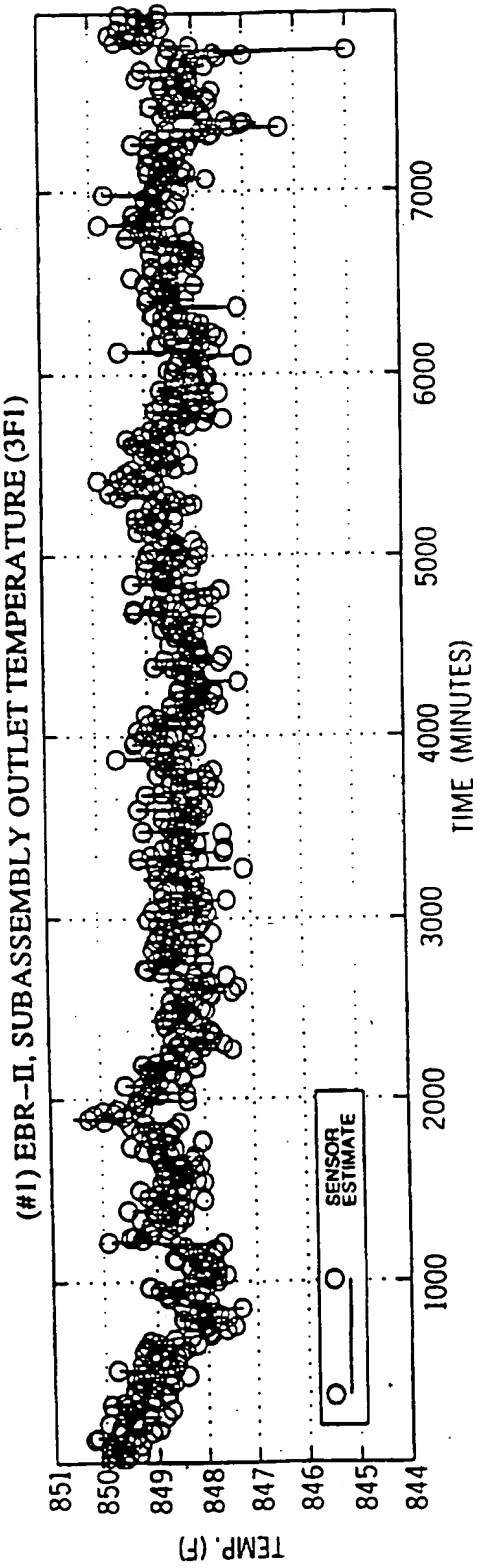

(#1) EBR-II, SUBASSEMBLY OUTLET TEMPERATURE (3F1)
TEMP. (F)
851
850
849
848
847
846
845
844
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)
SENSOR
ESTIMATE

FIG. 27A

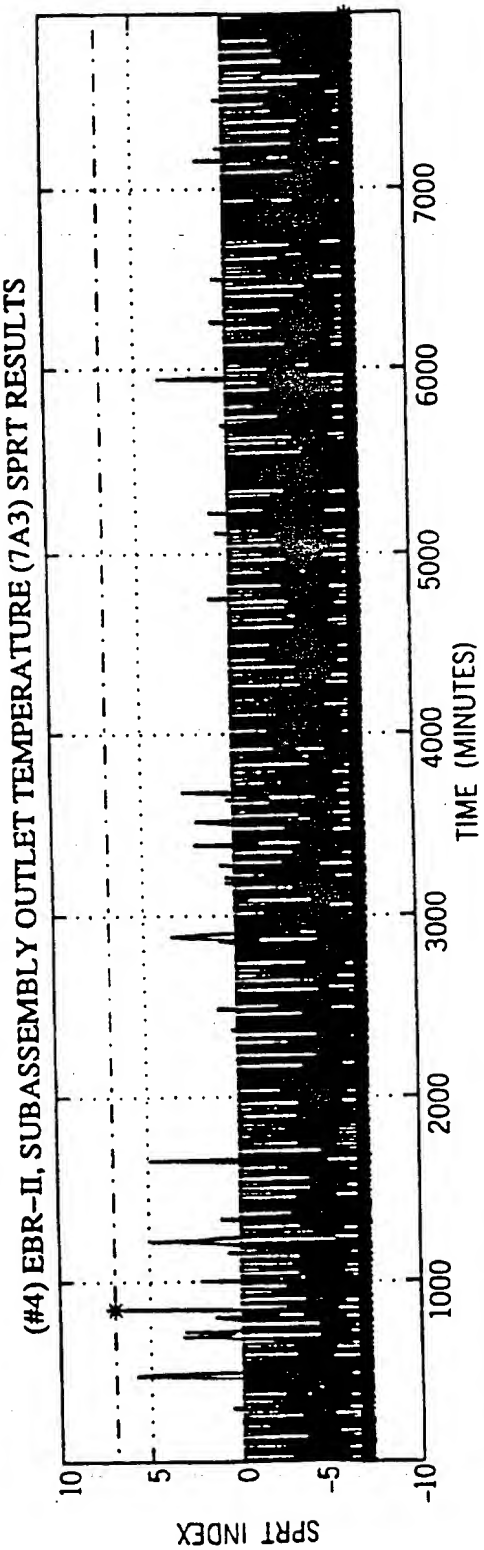

(#4) EBR-II, SUBASSEMBLY OUTLET TEMPERATURE (7A3) SPRT RESULTS
SPRT INDEX
10
5
0
-5
-10
TIME (MINUTES)
1000
2000
3000
4000
5000
6000
7000

FIG. 26D

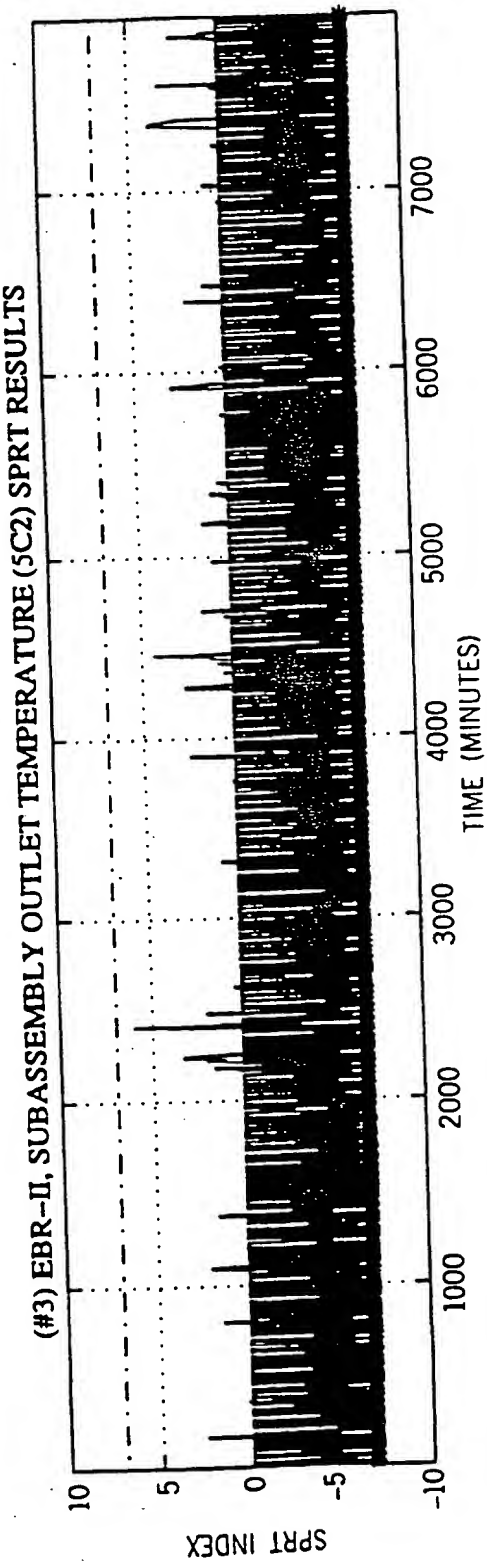

(#3) EBR-II, SUBASSEMBLY OUTLET TEMPERATURE (5C2) SPRT RESULTS
SPRT INDEX
10
5
0
-5
-10
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)

FIG. 26C

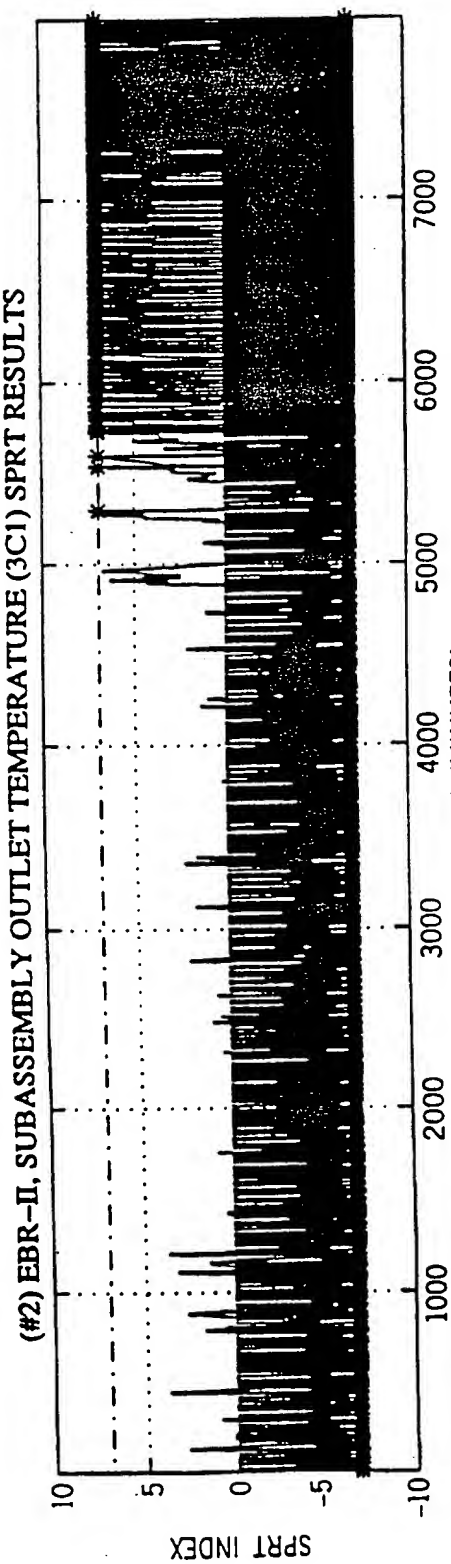

(#2) EBR-II, SUBASSEMBLY OUTLET TEMPERATURE (3C1) SPRT RESULTS
SPRT INDEX
10
5
0
-5
-10
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)

FIG. 26B

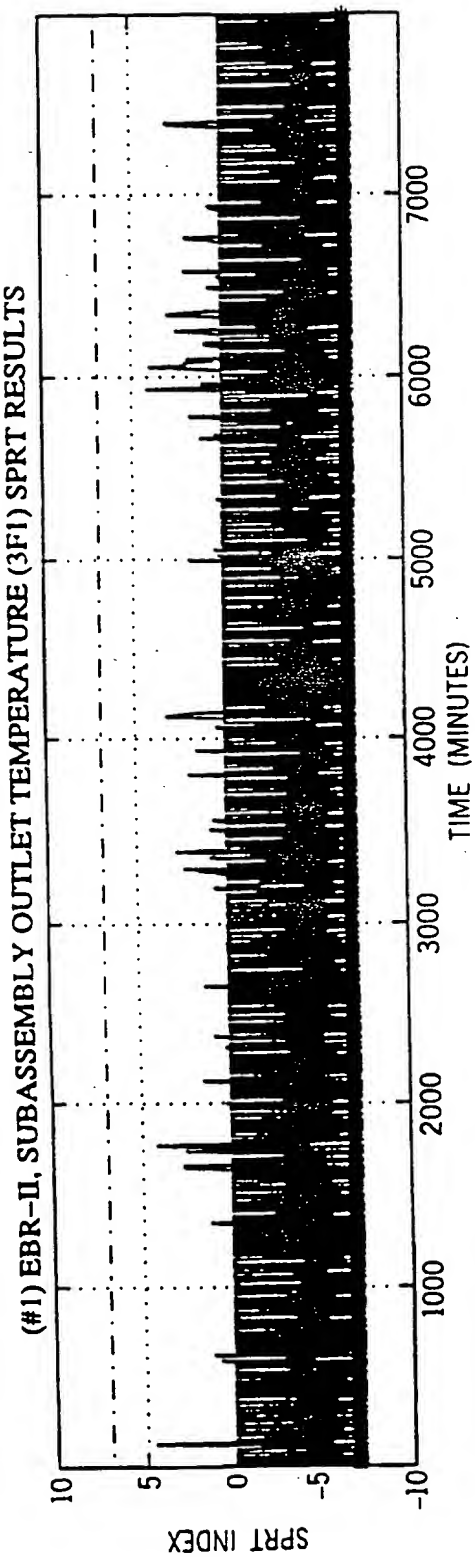

(#1) EBR-II, SUBASSEMBLY OUTLET TEMPERATURE (3F1) SPRT RESULTS
SPRT INDEX
10
5
0
-5
-10
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)

FIG. 26A

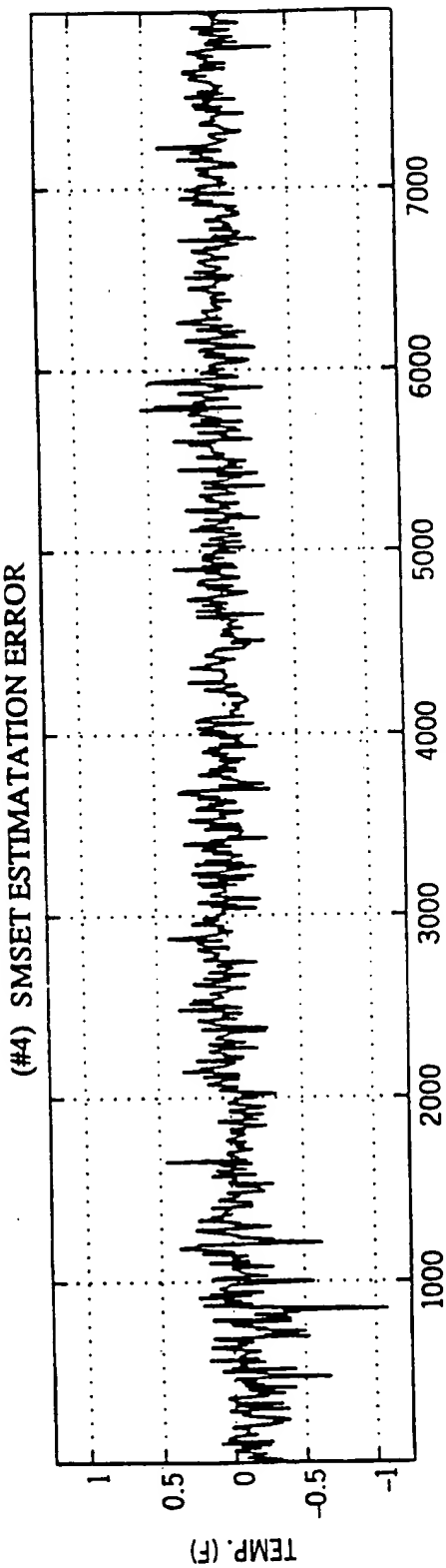

(#4) SMSET ESTIMATATION ERROR
TEMP. (F)
1
0.5
0
-0.5
-1
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)

FIG. 25D

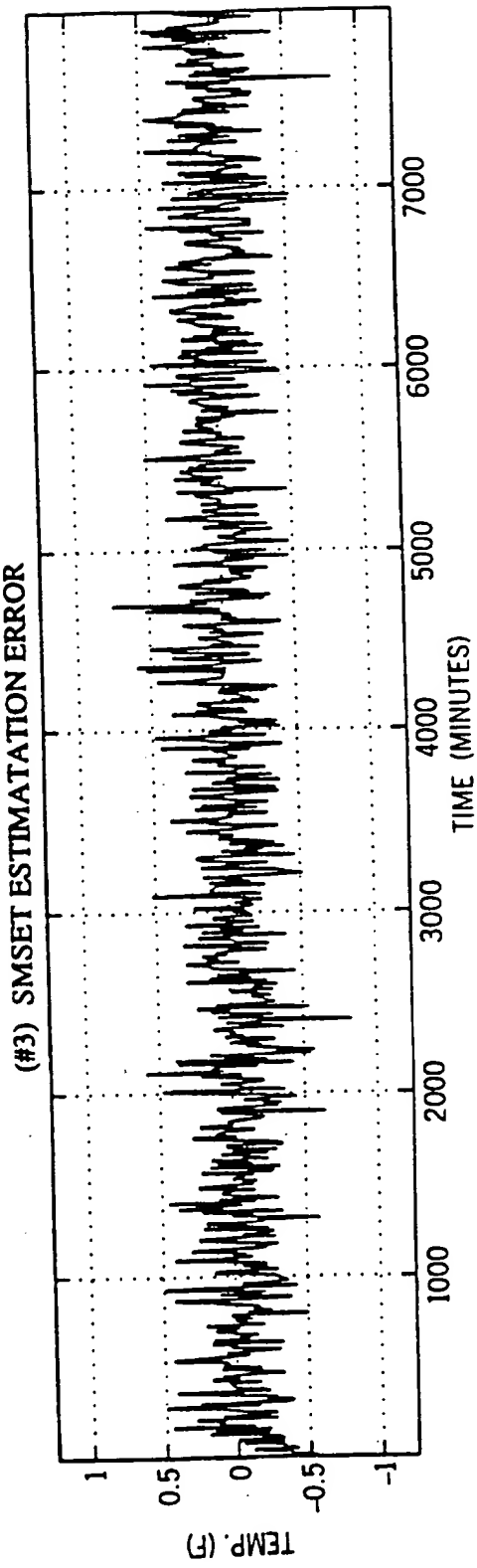

(#3) SMSET ESTIMATATION ERROR
TEMP. (F)
1
0.5
0
-0.5
-1
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)

FIG. 25C

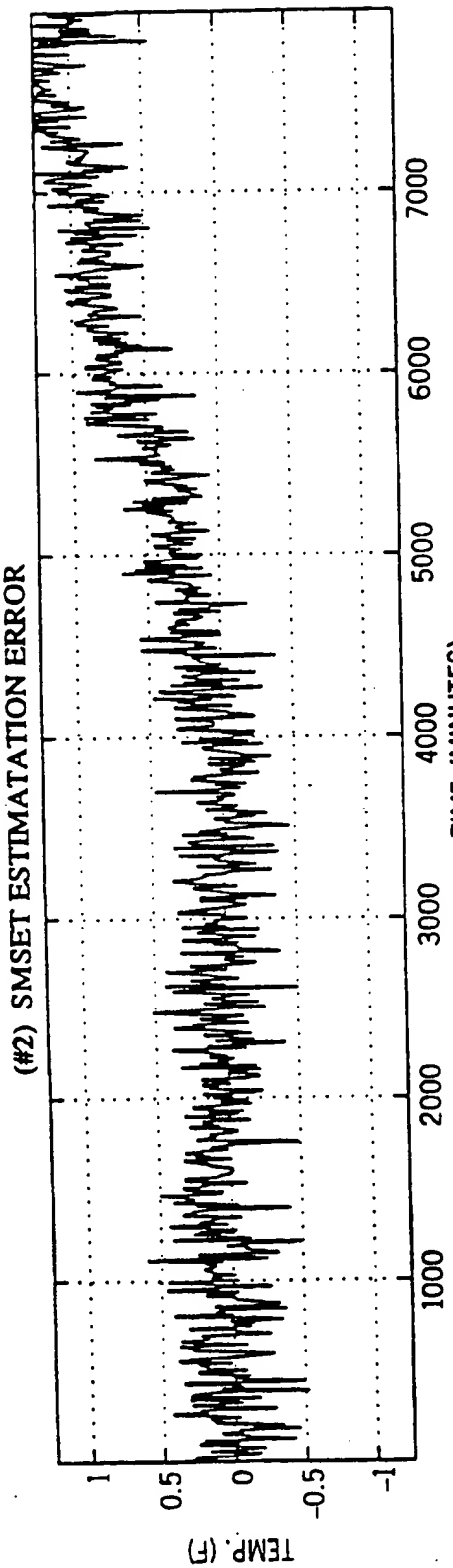

(#2) SMSET ESTIMATATION ERROR
TEMP. (F)
1
0.5
0
-0.5
-1
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)

FIG. 25B

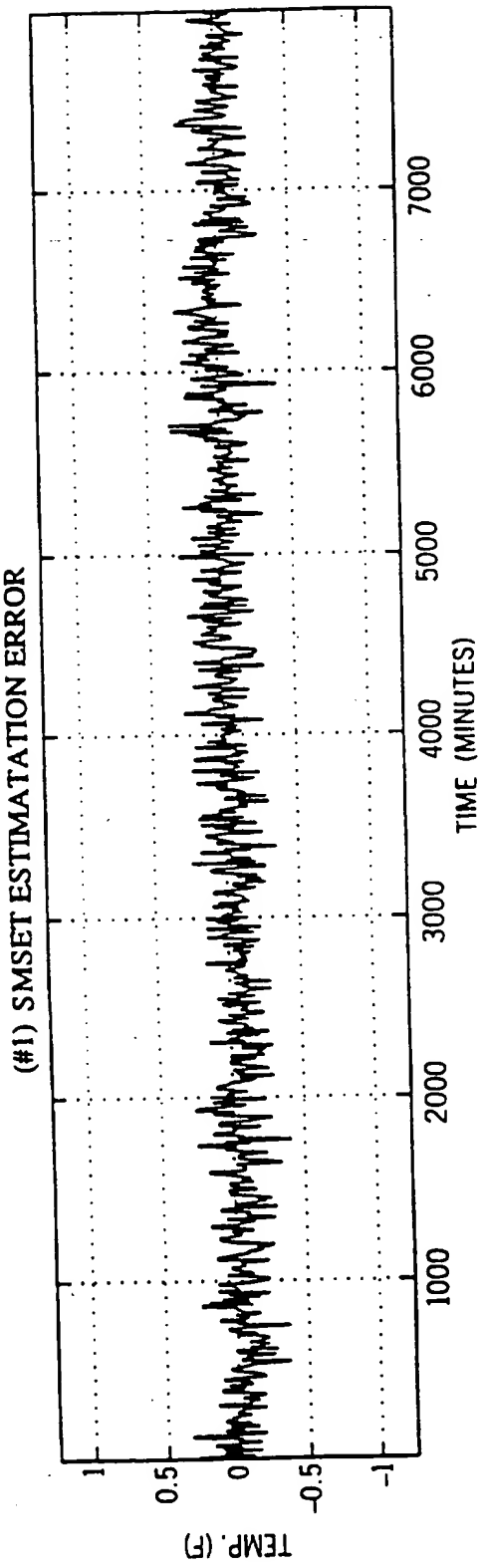

(#1) SMSET ESTIMATATION ERROR
TEMP. (F)
1
0.5
0
-0.5
-1
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)

FIG. 25A

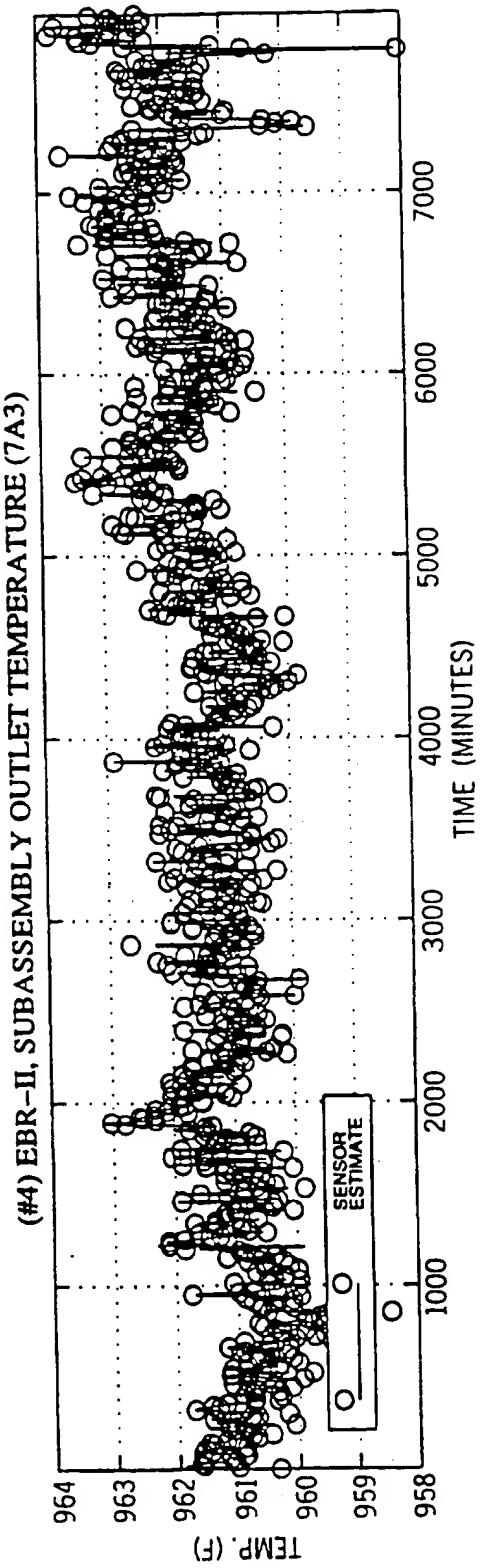

(#4) EBR-II, SUBASSEMBLY OUTLET TEMPERATURE (7A3)
TEMP. (F)
964
963
962
961
960
959
958
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)
SENSOR
ESTIMATE

FIG. 24D

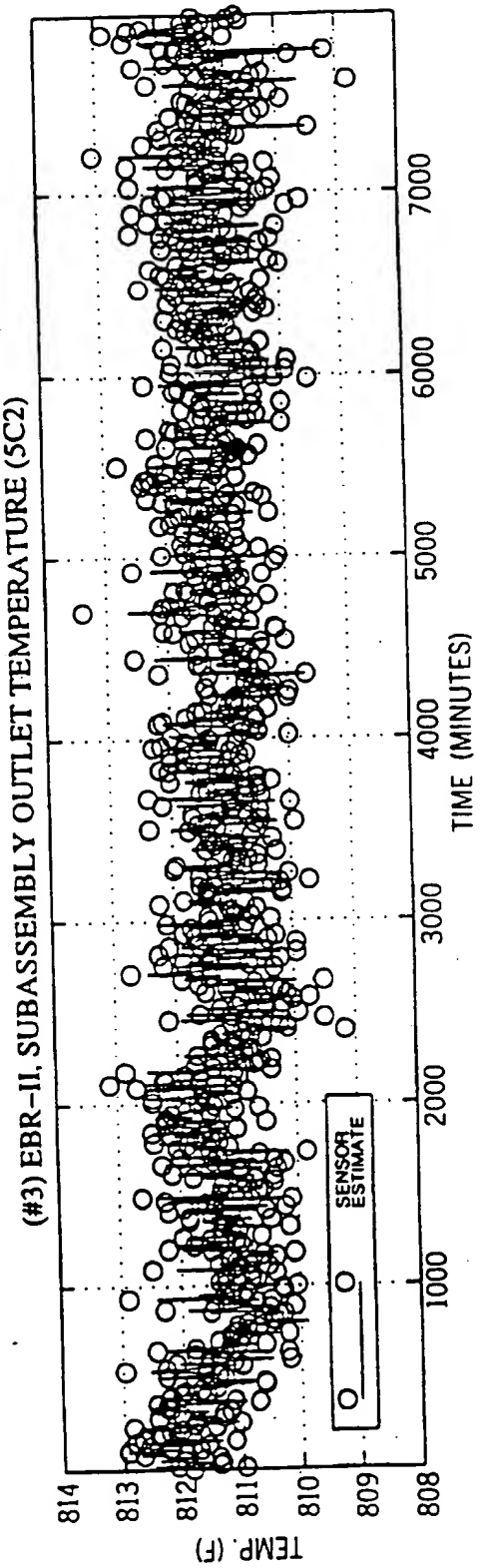

(#3) EBR-II, SUBASSEMBLY OUTLET TEMPERATURE (5C2)
TEMP. (F)
814
813
812
811
810
809
808
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)
SENSOR
ESTIMATE

FIG. 24C

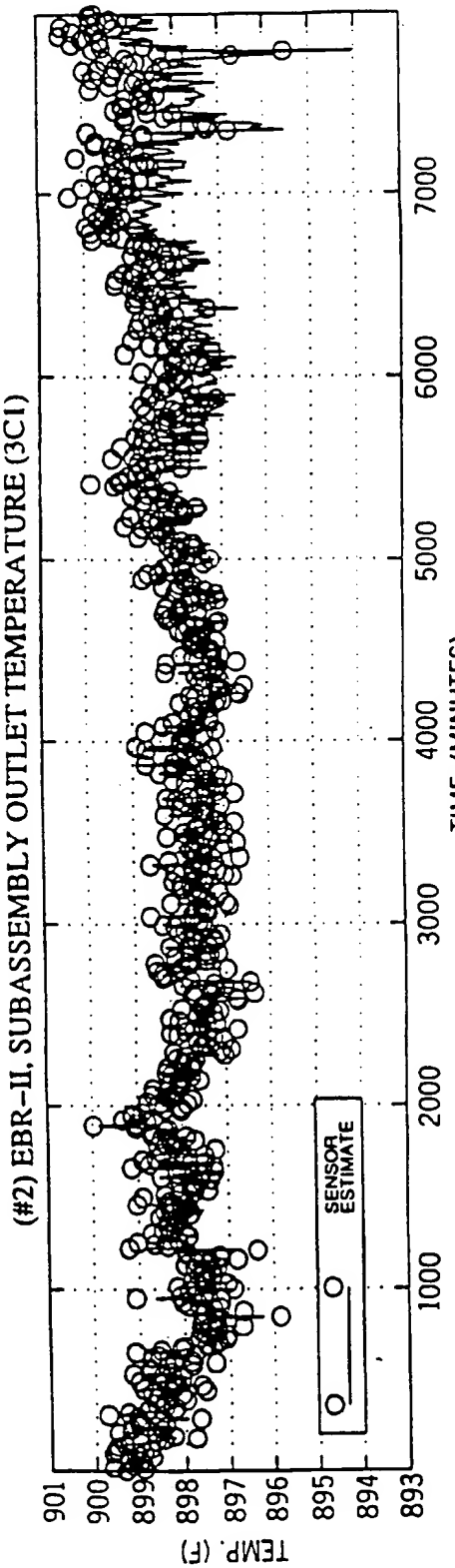

(#2) EBR-II, SUBASSEMBLY OUTLET TEMPERATURE (3C1)
TEMP. (F)
901
900
899
898
897
896
895
894
893
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)
SENSOR
ESTIMATE

FIG. 24B

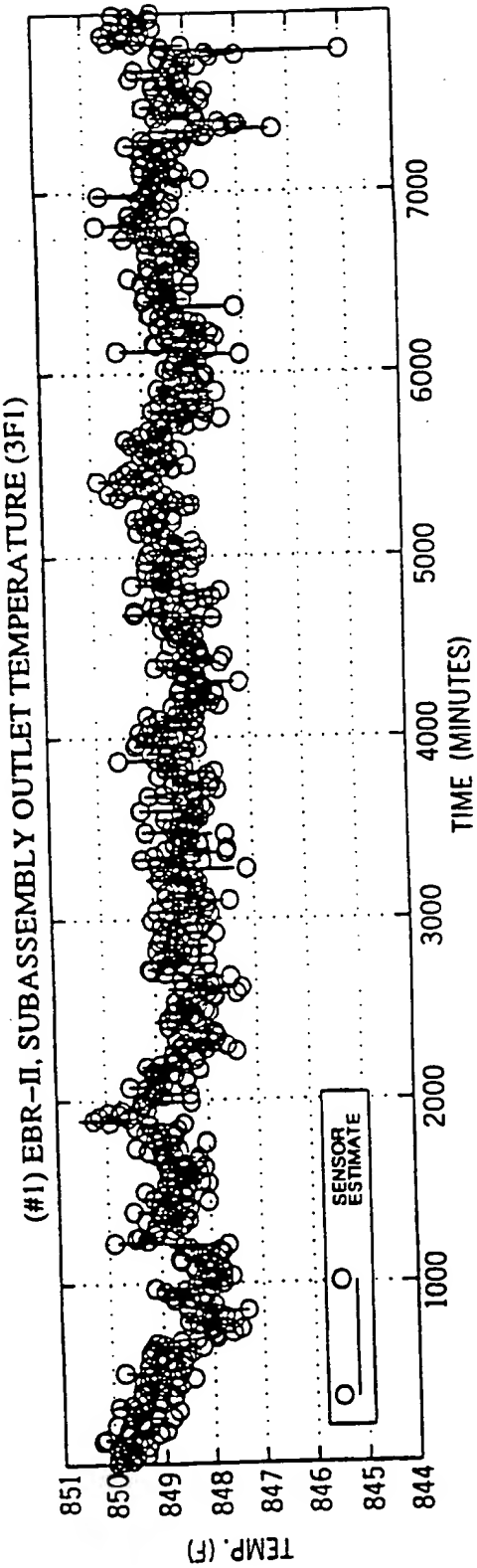

(#1) EBR-II, SUBASSEMBLY OUTLET TEMPERATURE (3F1)
TEMP. (F)
851
850
849
848
847
846
845
844
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)
SENSOR
ESTIMATE

FIG. 24A

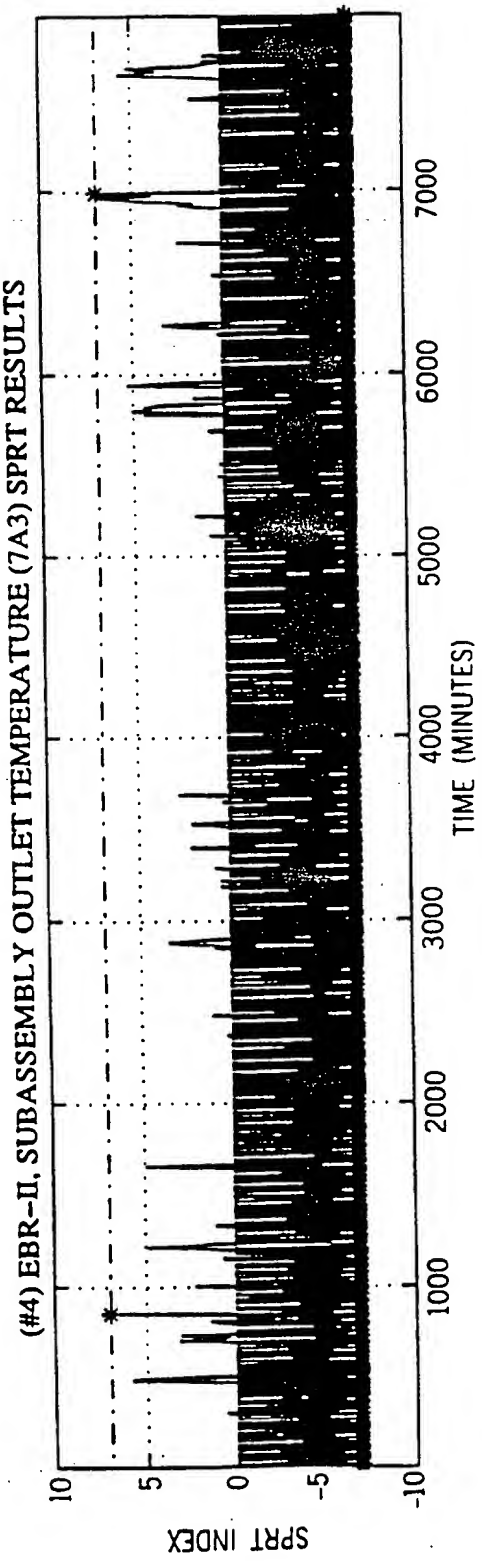

(#4) EBR-II, SUBASSEMBLY OUTLET TEMPERATURE (7A3) SPRT RESULTS
SPRT INDEX
10
5
0
-5
-10
TIME (MINUTES)
1000
2000
3000
4000
5000
6000
7000

FIG. 23D

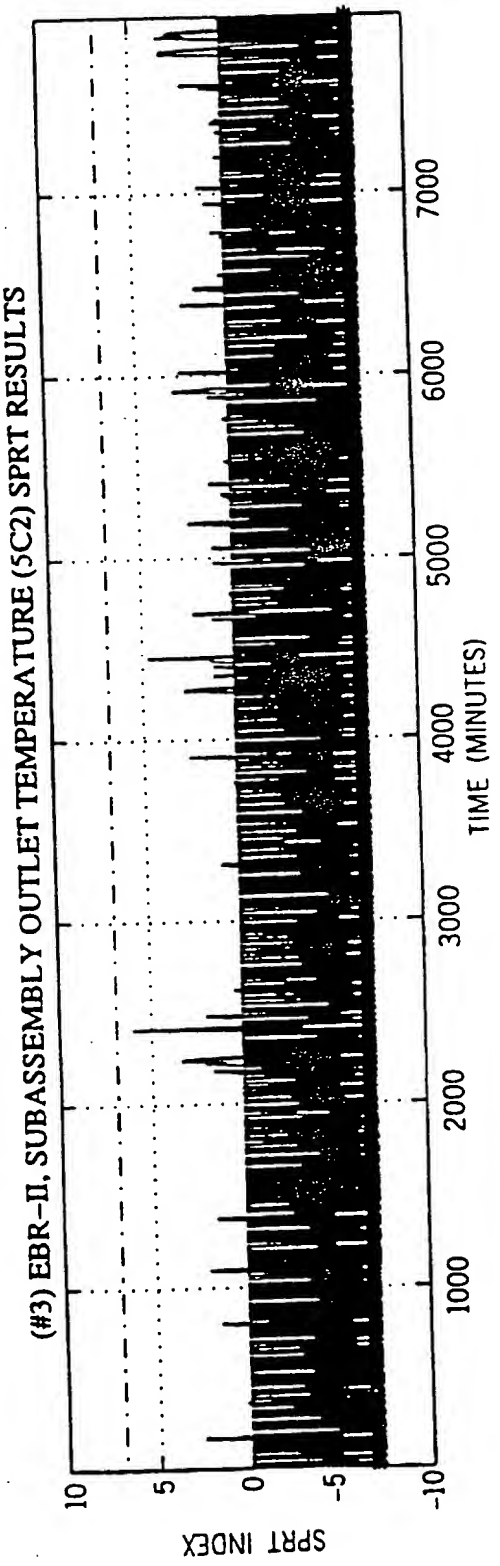

(#3) EBR-II, SUBASSEMBLY OUTLET TEMPERATURE (5C2) SPRT RESULTS
SPRT INDEX
10
5
0
-5
-10
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)

FIG. 23C

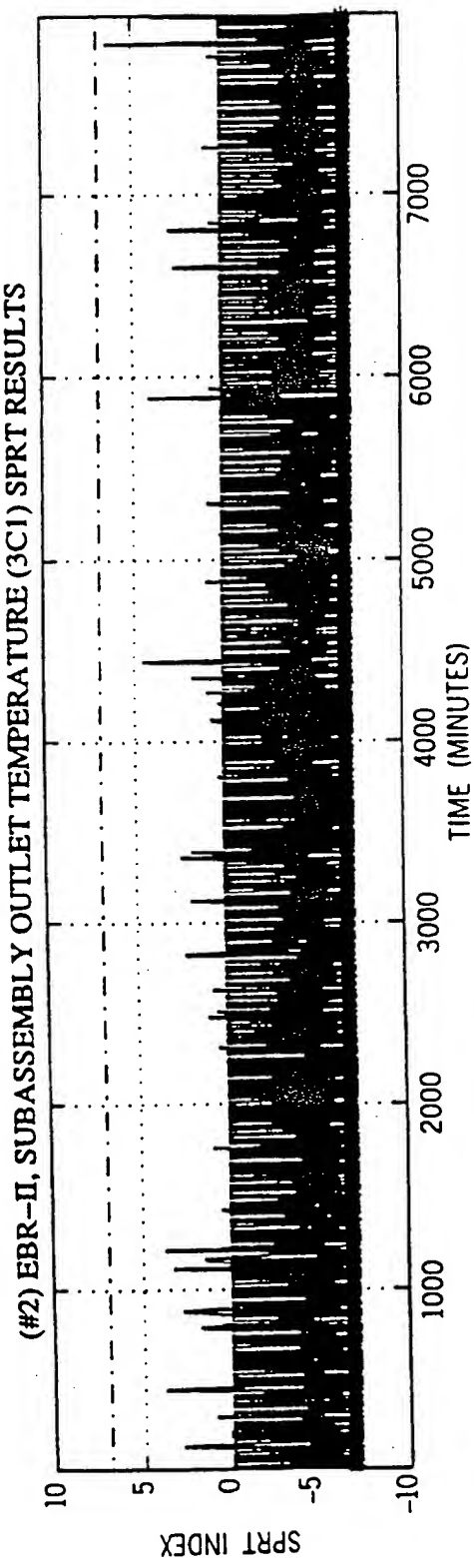

(#2) EBR-II, SUBASSEMBLY OUTLET TEMPERATURE (3C1) SPRT RESULTS
SPRT INDEX
10
5
0
-5
-10
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)

FIG. 23B

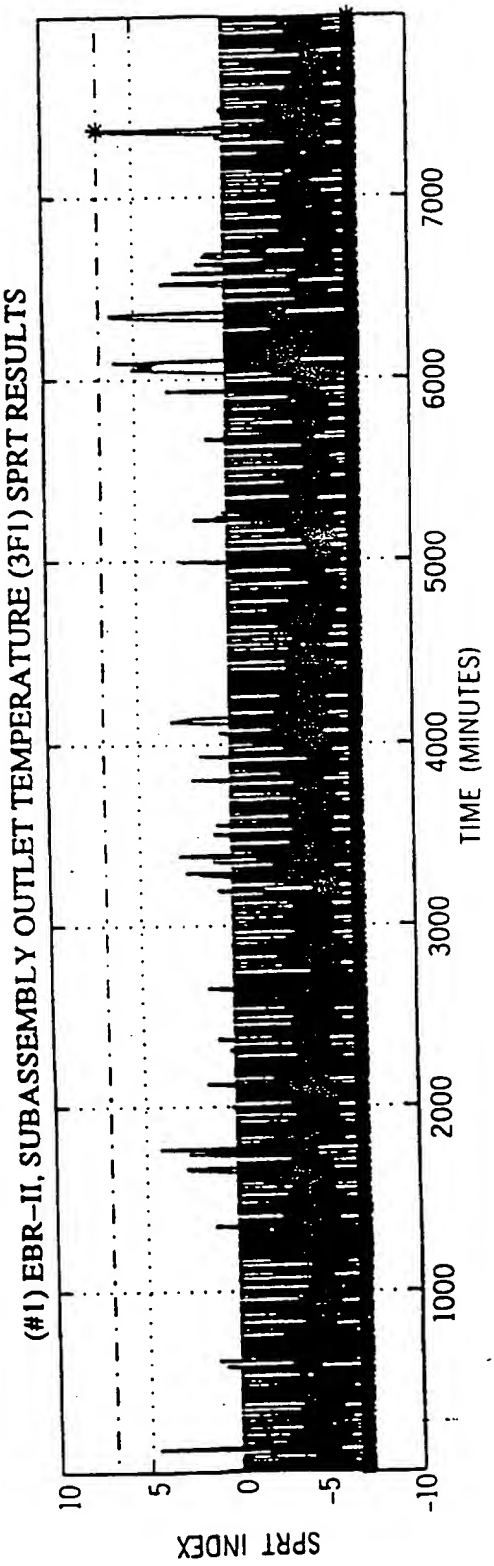

(#1) EBR-II, SUBASSEMBLY OUTLET TEMPERATURE (3F1) SPRT RESULTS
SPRT INDEX
10
5
0
-5
-10
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)

FIG. 23A

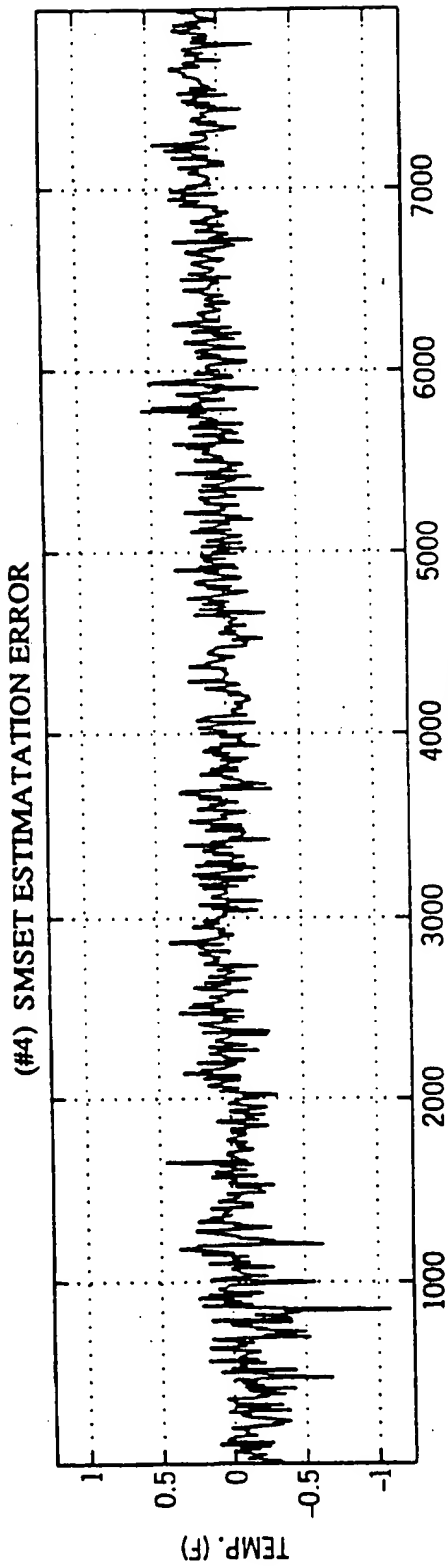

(#4) SMSET ESTIMATATION ERROR
TEMP. (F)
1
0.5
0
-0.5
-1
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)

FIG. 22D

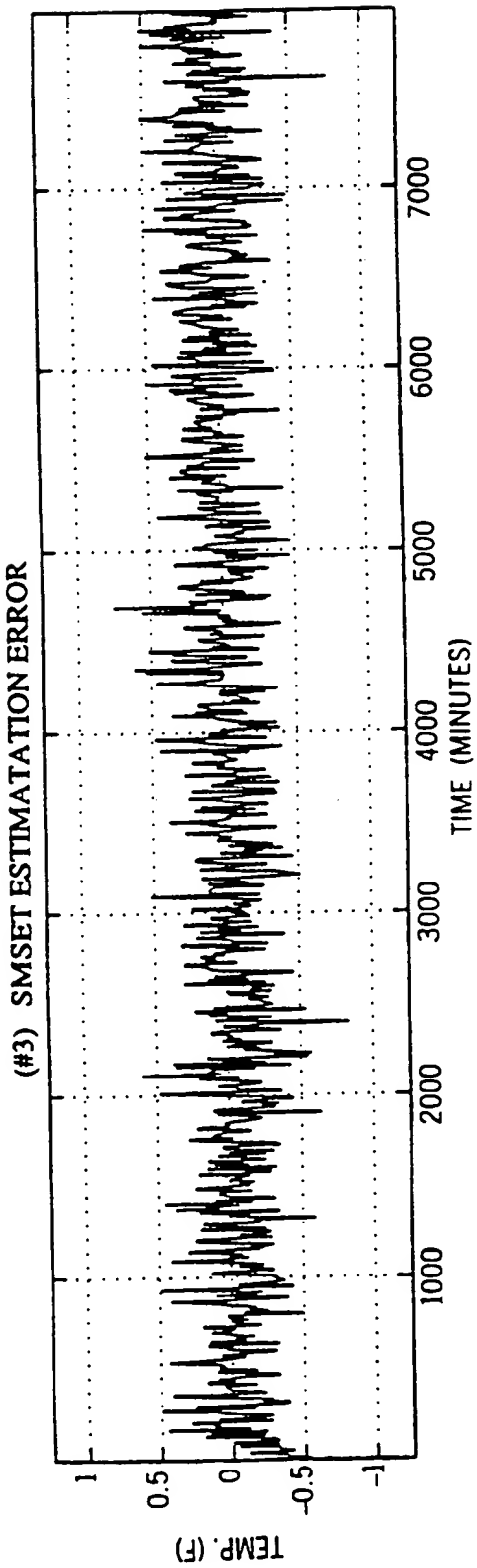

(#3) SMSET ESTIMATATION ERROR
TEMP. (F)
1
0.5
0
-0.5
-1
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)

FIG. 22C

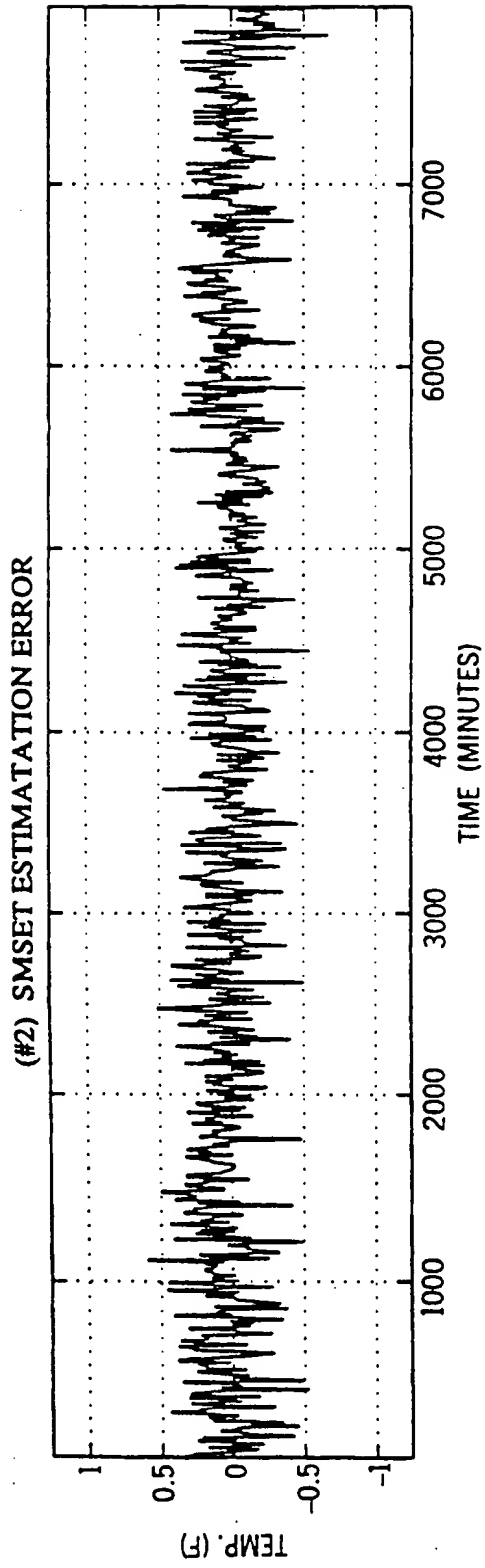

(#2) SMSET ESTIMATATION ERROR
TEMP. (F)
1
0.5
0
-0.5
-1
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)

FIG. 22B

AMENDED CLAIMS
[received by the International Bureau 24 October 1997 (24.10.97);
original claim 1 amended; new claims 17-30 added;
remaining claims unchanged (5 pages)]

1. A method for monitoring at least one of an industrial process and industrial sensors, comprising the steps of:

generating time varying data from a plurality of industrial sensors;

processing the time varying data to effectuate optimum time correlation of the data accumulated from the plurality of industrial sensors;

searching the time correlated data to identify maximum and minimum values for the data, thereby determining a full range of values for the data from the industrial process;

determining learned states of a normal operational condition of the industrial process and using the learned states to generate expected values of the operating industrial process;

comparing the expected values to current actual values of the industrial process to identify a current state of the industrial process closest to one of the learned states and generating a set of modeled data;

processing the modeled data to identify a pattern for the data and upon detecting a deviation from a pattern characteristics of normal operation, an alarm is generated.

2. The method as defined in Claim 1 wherein the industrial process comprises a physical process.

3. The method as defined in Claim 1 wherein the industrial process comprises a financial process.

4. The method as defined in Claim 1 wherein the step of determining optimum time correlation comprises comparing pairs of sensor signals, each characteristic of a separate sensor and calculating a cross correlation vector over time, applying a low pass filter to remove noise from the cross correlation vector and determining phase shift between the sensor signals.

17. A method for monitoring at least one of an industrial process and an individual date source, comprising the steps of:

sensing time varying data from at least one industrial data source of an industrial process;

determining learned states of a desired operational condition of the industrial process to use the learned states to generate expected values of the industrial process;

comparing the expected values to current sensed values of the industrial process to identify a current state of the industrial process closest to one of the learned states and generating data characteristic of the current state; and

processing the data that is characteristic of the current state to identify a pattern for the data and upon detecting a deviation from a pattern characteristic of the desired operational condition, a signal is generated indicating at least one of the industrial process and the industrial data source is not of the desired operational condition.

18. The method as defined in Claim 17 further including the step of searching the time varying data, before comparing the current actual values to the expected values, to identify minimum and maximum values for the data, thereby establishing a full range of values for the data.

19. The method as defined in Claim 18 wherein each said industrial data source is characterized by two data values associated with the minimum and maximum values.

20. The method as defined in Claim 17 wherein said step of processing the data characteristic of the current state to identify a pattern comprises applying a sequential probability ratio test.

21. The method as defined in claim 20 wherein data that is characteristic of the current state is processed to generate a set of modeled data which is further processed to identify the pattern for the data.

22. The method as defined in Claim 17 wherein the industrial process is selected from the group consisting of a manufacturing process, a physical process, a chemical process, a biological process, an electronic process and a financial process.

23. The method as defined in Claim 17 further including the step of substituting an estimated signal for said industrial data source upon detecting the deviation from a pattern characteristic of the desired operational condition, thereby replacing a faulted data source enabling continued operation and monitoring of the industrial process.

27. The method as defined in Claim 26 further including the step of substituting an estimated signal for said industrial data source upon detecting the deviation from a pattern and characteristic of the desired operational condition, thereby replacing a faulted data source enabling continued operation and monitoring of the industrial process.

28. The method as defined in Claim 26 further including the step of processing the time varying data to effectuate optimum time correlation.

29. The method as defined in Claim 26 wherein the step of identifying a pattern for the time varying data comprises applying a sequential probability ratio test.

30. The method as defined in Claim 26 wherein the industrial process is selected from the group consisting of a manufacturing process a physical process, a chemical process, a financial process, an electronic process and a biological process.

1/96

FIG. 1

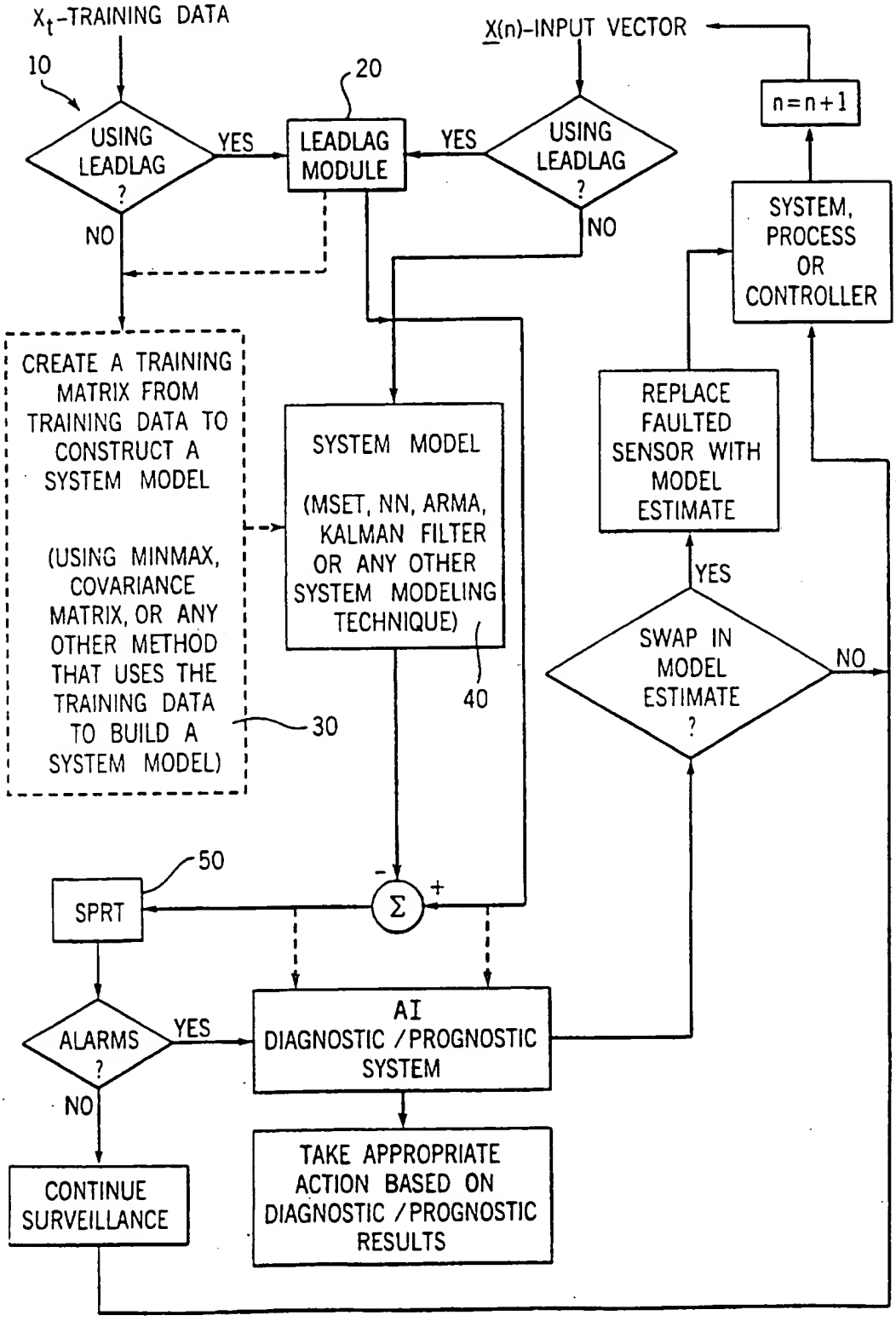

X_t-TRAINING DATA
10
USING LEADLAG ?
YES
20
LEADLAG MODULE
YES
USING LEADLAG ?
X(n)-INPUT VECTOR
n=n+1
NO
NO
SYSTEM, PROCESS OR CONTROLLER
CREATE A TRAINING MATRIX FROM TRAINING DATA TO CONSTRUCT A SYSTEM MODEL
(USING MINMAX, COVARIANCE MATRIX, OR ANY OTHER METHOD THAT USES THE TRAINING DATA TO BUILD A SYSTEM MODEL)
30
SYSTEM MODEL
(MSET, NN, ARMA, KALMAN FILTER OR ANY OTHER SYSTEM MODELING TECHNIQUE)
40
REPLACE FAULTED SENSOR WITH MODEL ESTIMATE
YES
SWAP IN MODEL ESTIMATE ?
NO
50
SPRT
-
Σ
+
ALARMS ?
YES
AI DIAGNOSTIC / PROGNOSTIC SYSTEM
NO
CONTINUE SURVEILLANCE
TAKE APPROPRIATE ACTION BASED ON DIAGNOSTIC / PROGNOSTIC RESULTS

FIG. 2

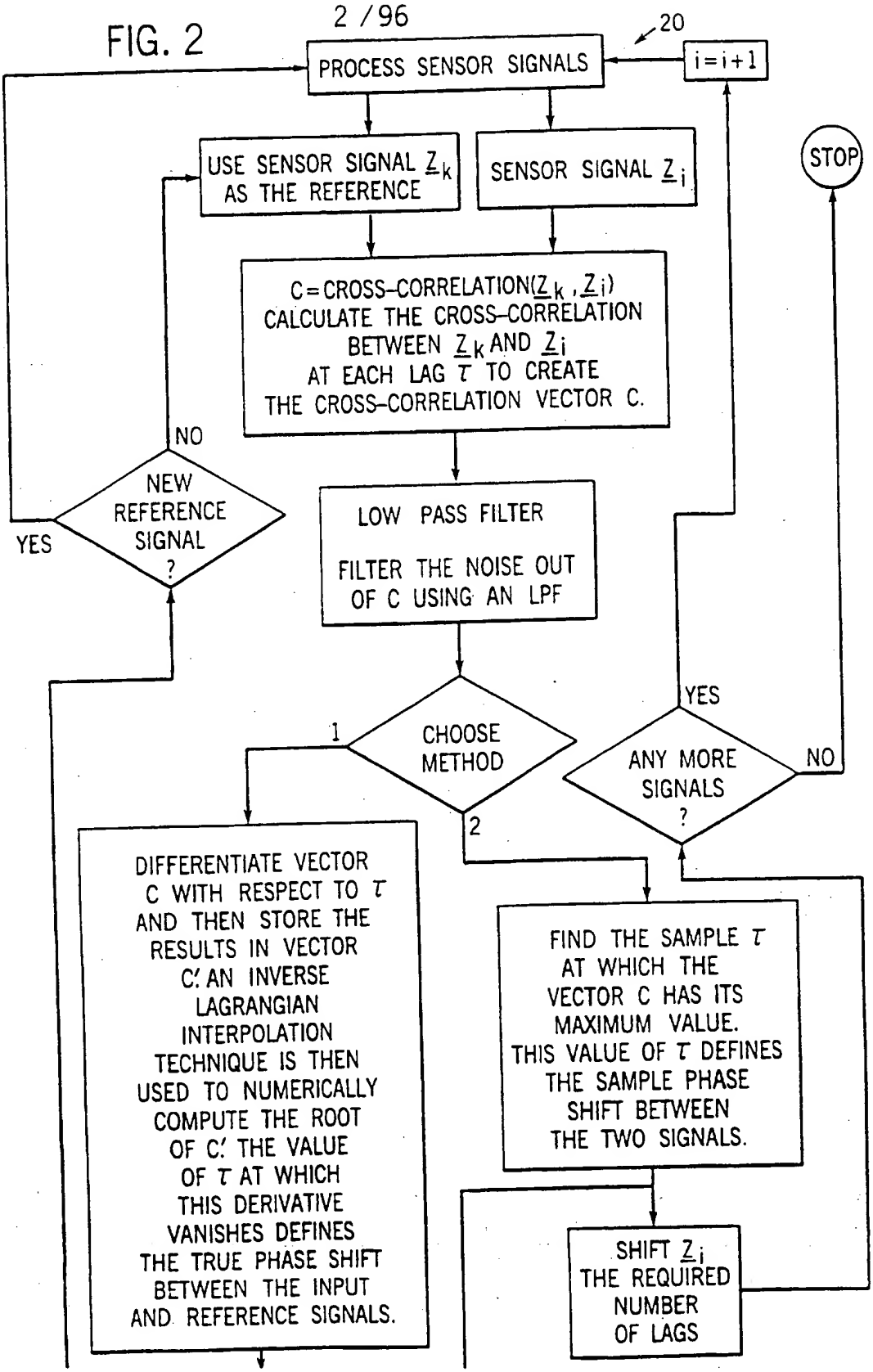

20
PROCESS SENSOR SIGNALS
i=i+1
USE SENSOR SIGNAL Z_k AS THE REFERENCE
SENSOR SIGNAL Z_i
STOP
C=CROSS-CORRELATION(Z_k, Z_i)
CALCULATE THE CROSS-CORRELATION BETWEEN Z_k AND Z_i AT EACH LAG τ TO CREATE THE CROSS-CORRELATION VECTOR C.
NO
NEW REFERENCE SIGNAL ?
YES
LOW PASS FILTER
FILTER THE NOISE OUT OF C USING AN LPF
YES
1
CHOOSE METHOD
ANY MORE SIGNALS ?
NO
2
DIFFERENTIATE VECTOR C WITH RESPECT TO τ AND THEN STORE THE RESULTS IN VECTOR C'. AN INVERSE LAGRANGIAN INTERPOLATION TECHNIQUE IS THEN USED TO NUMERICALLY COMPUTE THE ROOT OF C'. THE VALUE OF τ AT WHICH THIS DERIVATIVE VANISHES DEFINES THE TRUE PHASE SHIFT BETWEEN THE INPUT AND REFERENCE SIGNALS.
FIND THE SAMPLE τ AT WHICH THE VECTOR C HAS ITS MAXIMUM VALUE. THIS VALUE OF τ DEFINES THE SAMPLE PHASE SHIFT BETWEEN THE TWO SIGNALS.
SHIFT Z_i THE REQUIRED NUMBER OF LAGS

FIG. 3

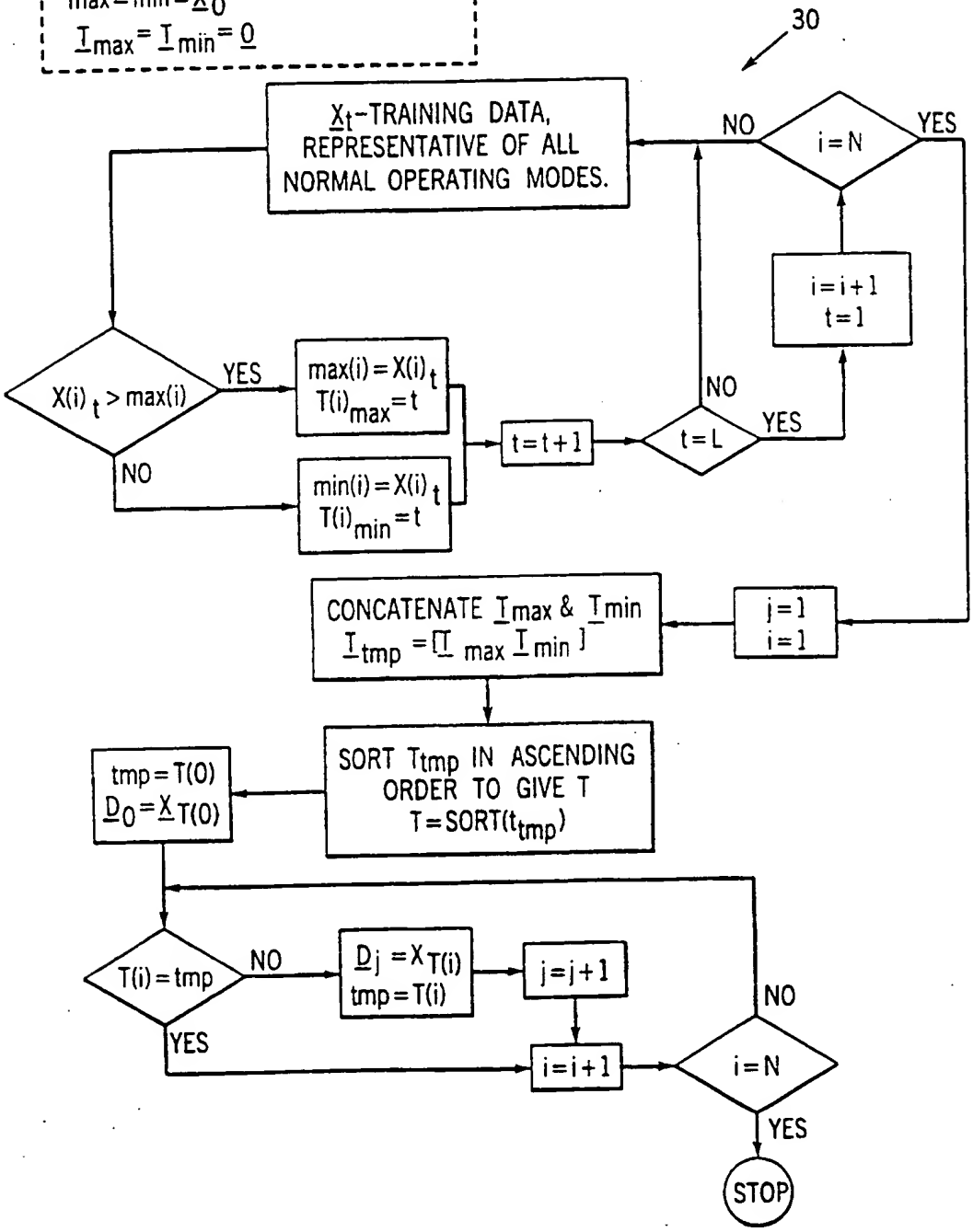
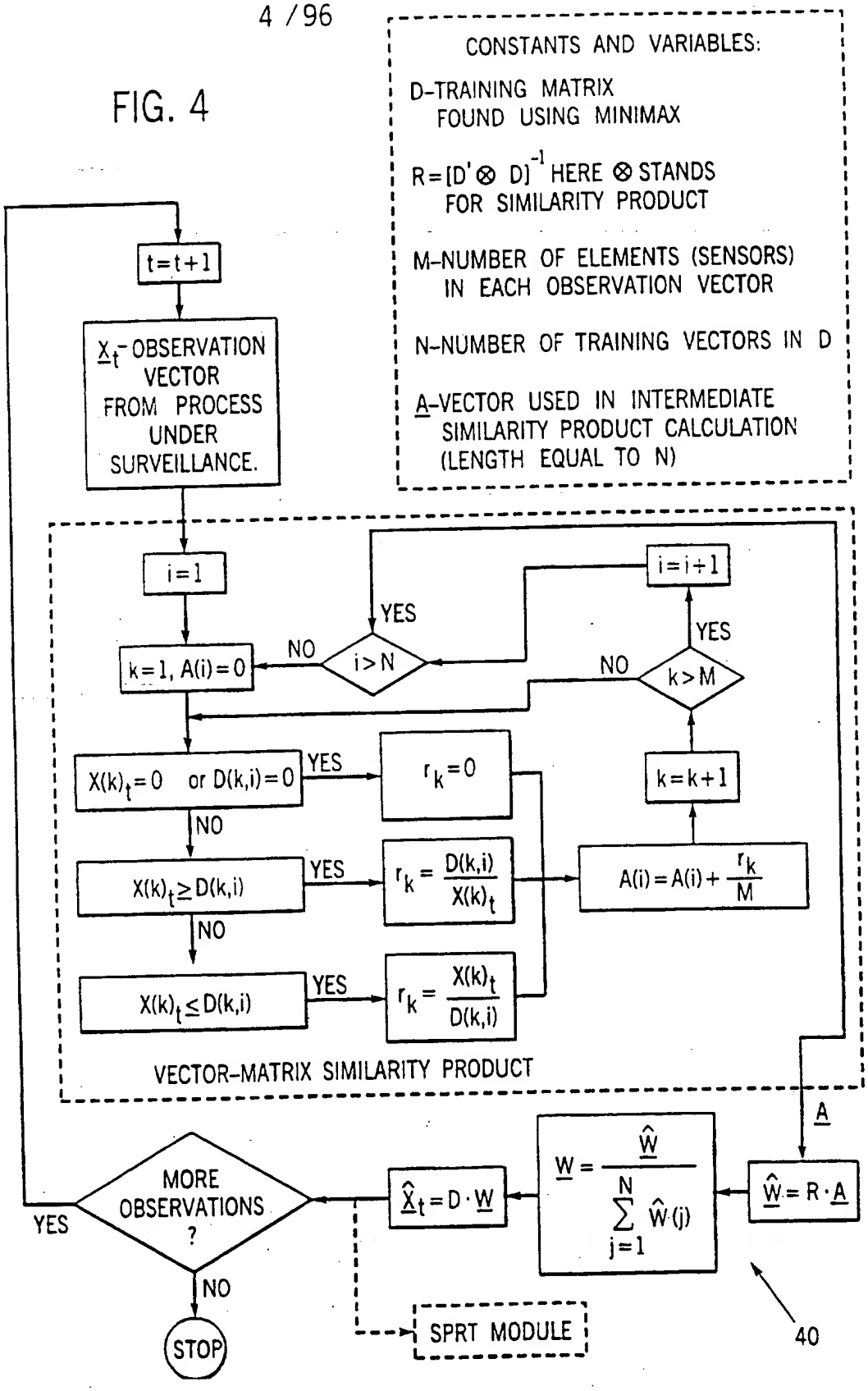

VARIABLES:
N-NUMBERS OF SENSOR SIGNALS
L-NUMBER OF OBSERVATIONS
D-MINIMAX TRAINING MATRIX
INITIALIZE:
i=0: ELEMENT NUMBER
t=1: OBSERVATION NUMBER
max=min=X0
Imax=Imin=0
30
Xt-TRAINING DATA, REPRESENTATIVE OF ALL NORMAL OPERATING MODES.
NO
i=N
YES
i=i+1
t=1
X(i)t > max(i)
YES
max(i)=X(i)t
T(i)max=t
NO
min(i)=X(i)t
T(i)min=t
t=t+1
t=L
NO
YES
CONCATENATE Imax & Imin
Itmp=[I max Imin]
j=1
i=1
SORT Ttmp IN ASCENDING ORDER TO GIVE T
T=SORT(ttmp)
tmp=T(0)
D0=XT(0)
T(i)=tmp
NO
Dj=XT(i)
tmp=T(i)
j=j+1
YES
i=i+1
i=N
NO
YES
STOP

FIG. 4

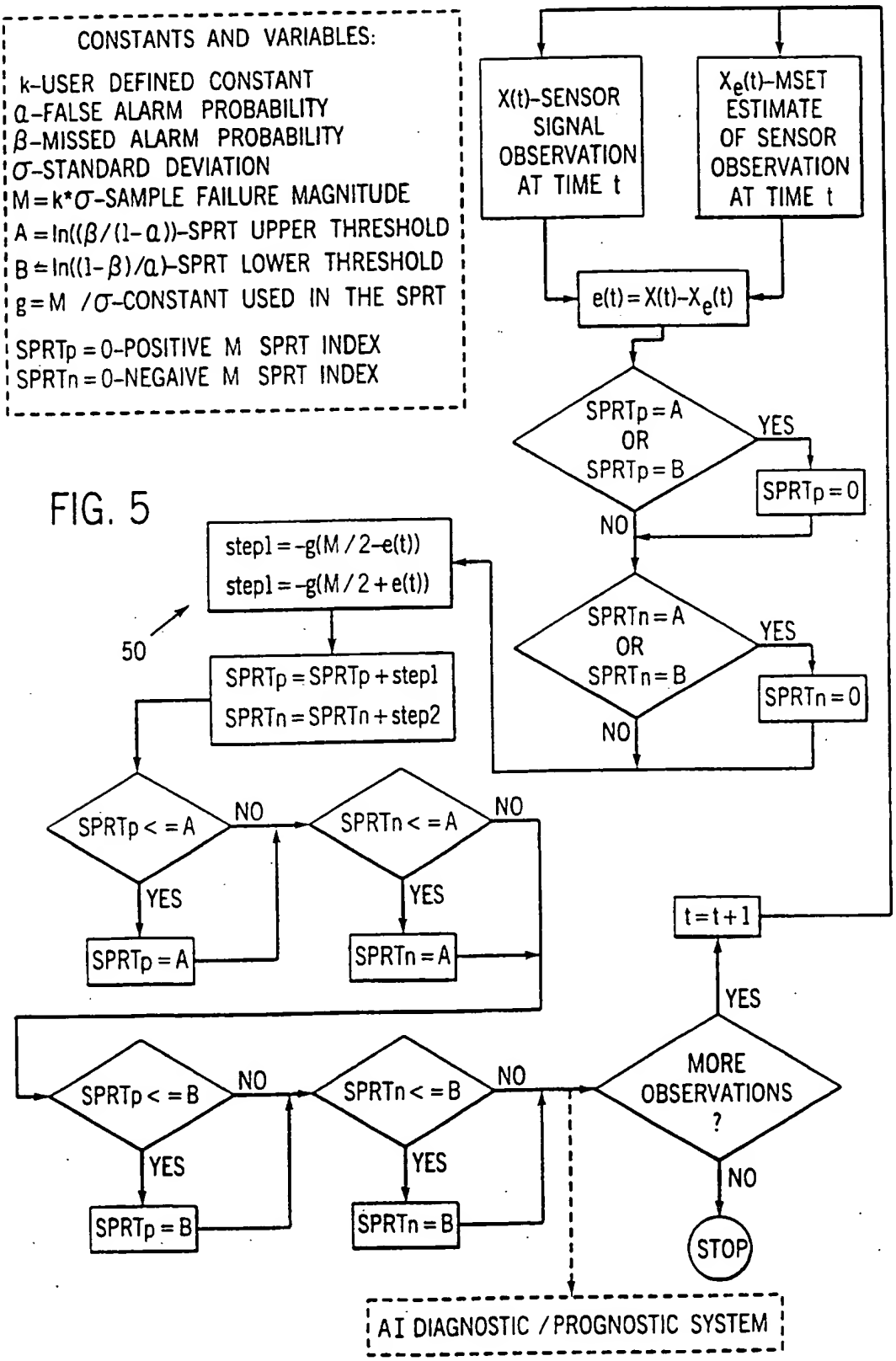

CONSTANTS AND VARIABLES:
D-TRAINING MATRIX FOUND USING MINIMAX
R=[D' ⊗ D]^-1 HERE ⊗ STANDS FOR SIMILARITY PRODUCT
M-NUMBER OF ELEMENTS (SENSORS) IN EACH OBSERVATION VECTOR
N-NUMBER OF TRAINING VECTORS IN D
A-VECTOR USED IN INTERMEDIATE SIMILARITY PRODUCT CALCULATION (LENGTH EQUAL TO N)
t=t+1
X_t- OBSERVATION VECTOR FROM PROCESS UNDER SURVEILLANCE.
i=1
k=1, A(i)=0
i>N
YES
NO
i=i+1
k>M
YES
NO
X(k)_t=0 or D(k,i)=0
YES
r_k=0
k=k+1
NO
X(k)_t≥D(k,i)
YES
r_k = D(k,i)/X(k)_t
A(i)=A(i)+r_k/M
NO
X(k)_t≤D(k,i)
YES
r_k = X(k)_t/D(k,i)
VECTOR-MATRIX SIMILARITY PRODUCT
A
W = Ŵ / Σ_{j=1}^{N} Ŵ(j)
Ŵ = R·A
X̂_t = D·W
MORE OBSERVATIONS ?
YES
NO
STOP
SPRT MODULE
40

CONSTANTS AND VARIABLES:

k-USER DEFINED CONSTANT
α-FALSE ALARM PROBABILITY
β-MISSED ALARM PROBABILITY
σ-STANDARD DEVIATION
M = k*σ-SAMPLE FAILURE MAGNITUDE
A = ln((β/(1-α))-SPRT UPPER THRESHOLD
B = ln((1-β)/α)-SPRT LOWER THRESHOLD
g = M /σ-CONSTANT USED IN THE SPRT

SPRTp = 0-POSITIVE M SPRT INDEX
SPRTn = 0-NEGAIVE M SPRT INDEX

FIG. 5



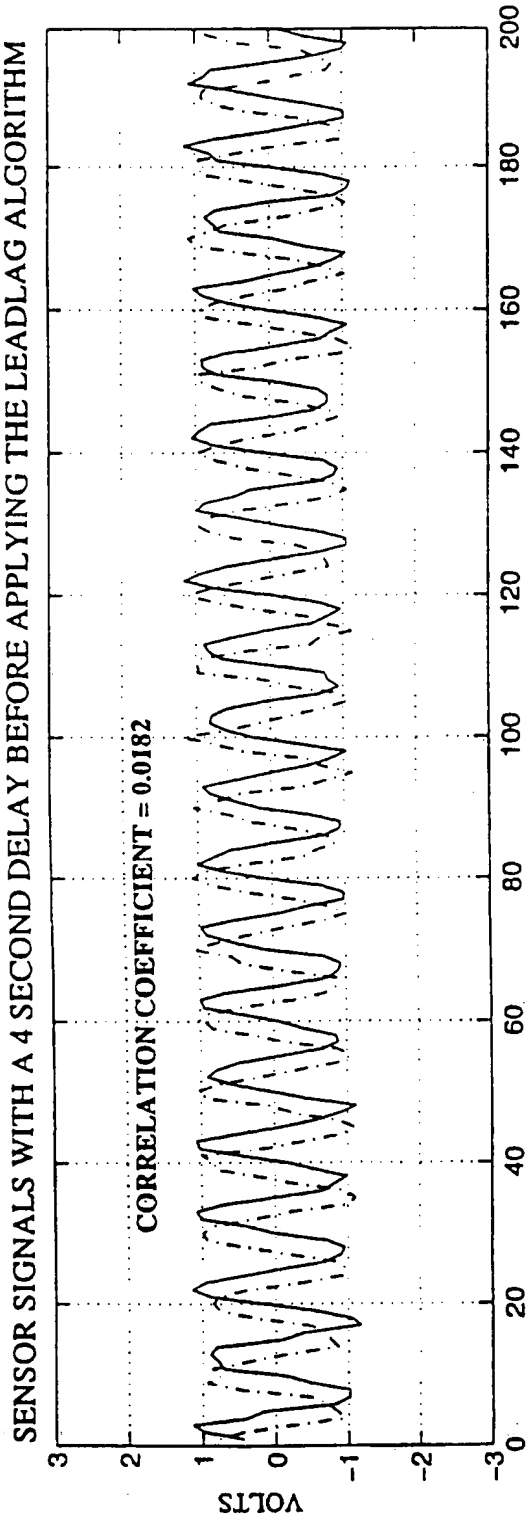

SENSOR SIGNALS WITH A 4 SECOND DELAY BEFORE APPLYING THE LEADLAG ALGORITHM
CORRELATION COEFFICIENT = 0.0182
VOLTS
3
2
1
0
-1
-2
-3
0
20
40
60
80
100
120
140
160
180
200

FIG. 6A

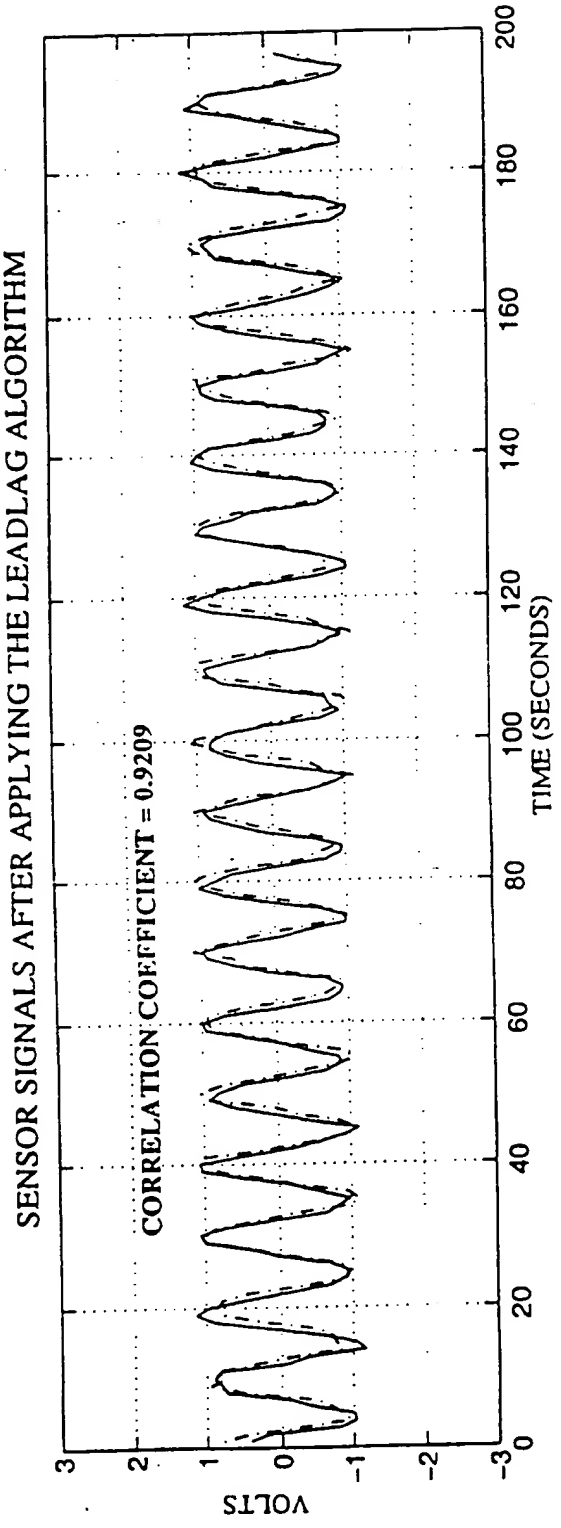

SENSOR SIGNALS AFTER APPLYING THE LEADLAG ALGORITHM
CORRELATION COEFFICIENT = 0.9209
VOLTS
TIME (SECONDS)
3
2
1
0
-1
-2
-3
0
20
40
60
80
100
120
140
160
180
200

FIG. 6B

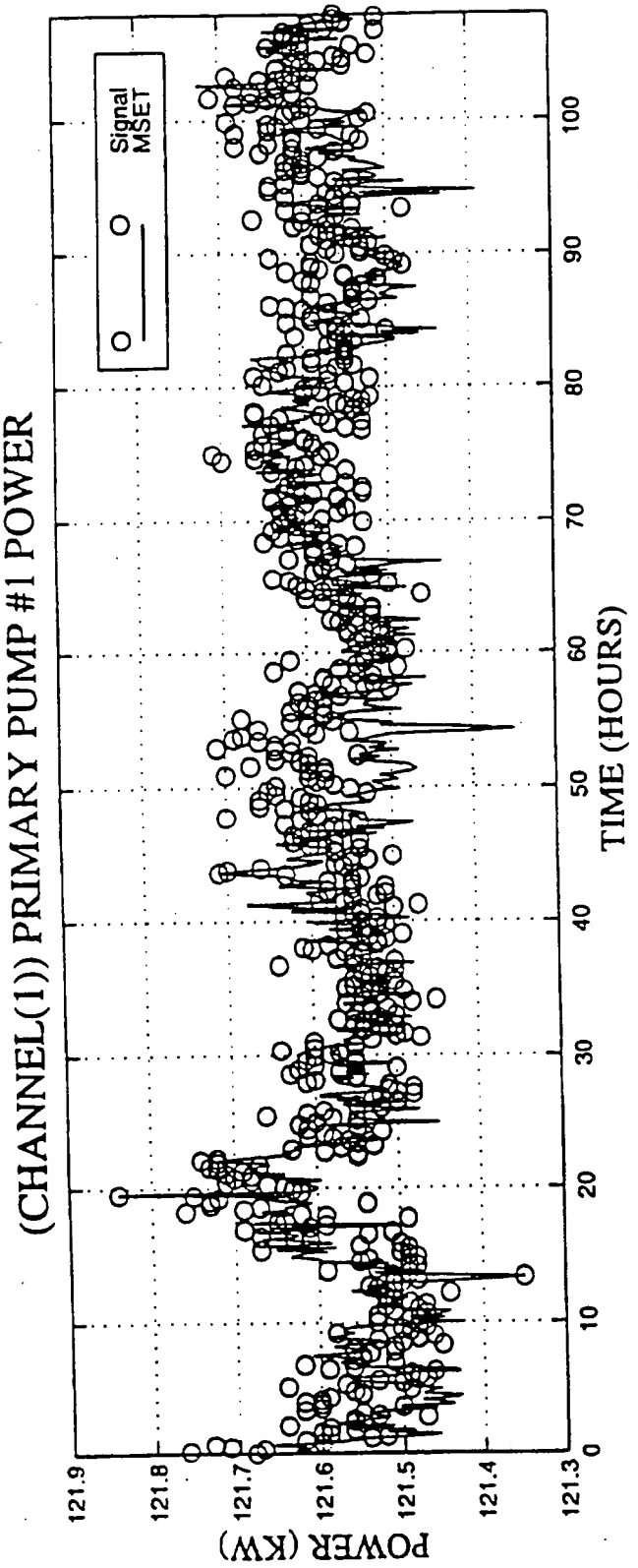

(CHANNEL(1)) PRIMARY PUMP #1 POWER
Signal
MSET
POWER (KW)
121.9
121.8
121.7
121.6
121.5
121.4
121.3
TIME (HOURS)
0
10
20
30
40
50
60
70
80
90
100

FIG. 7A

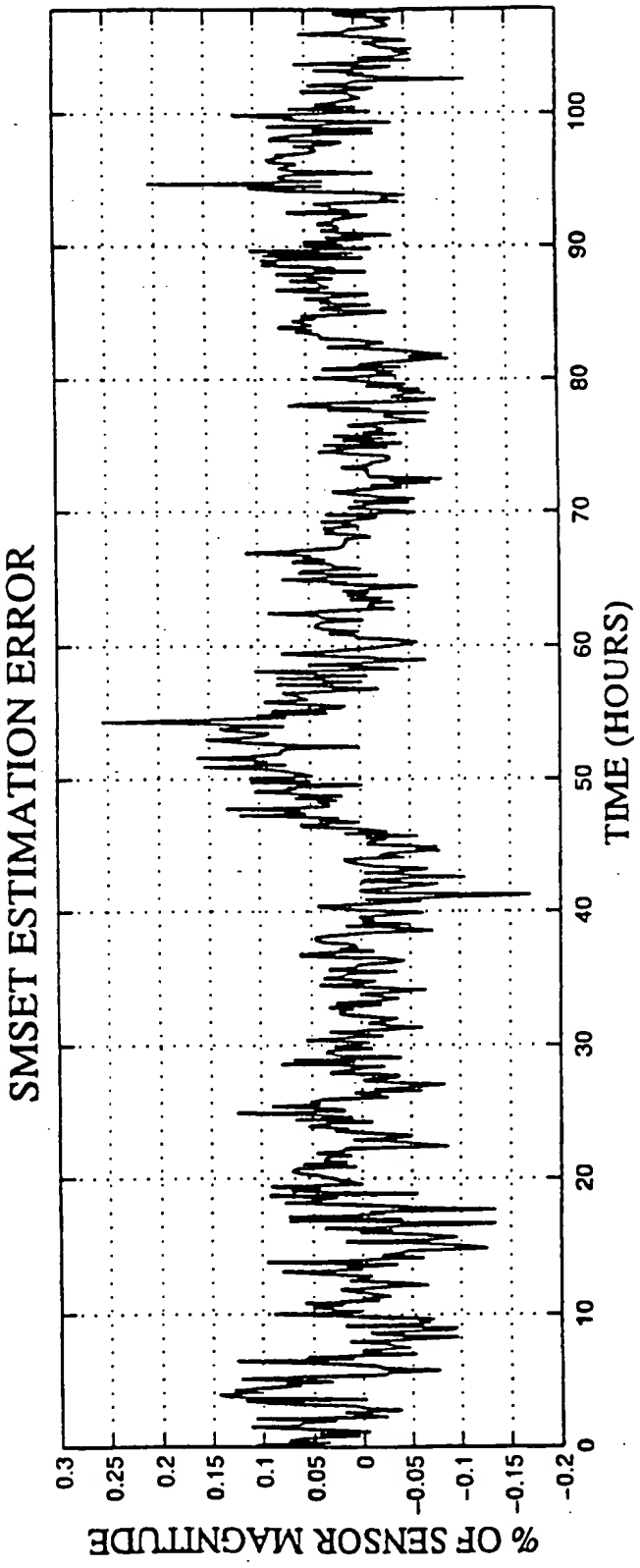

SMSET ESTIMATION ERROR
% OF SENSOR MAGNITUDE
0.3
0.25
0.2
0.15
0.1
0.05
0
−0.05
−0.1
−0.15
−0.2
0
10
20
30
40
50
60
70
80
90
100
TIME (HOURS)

FIG. 7B

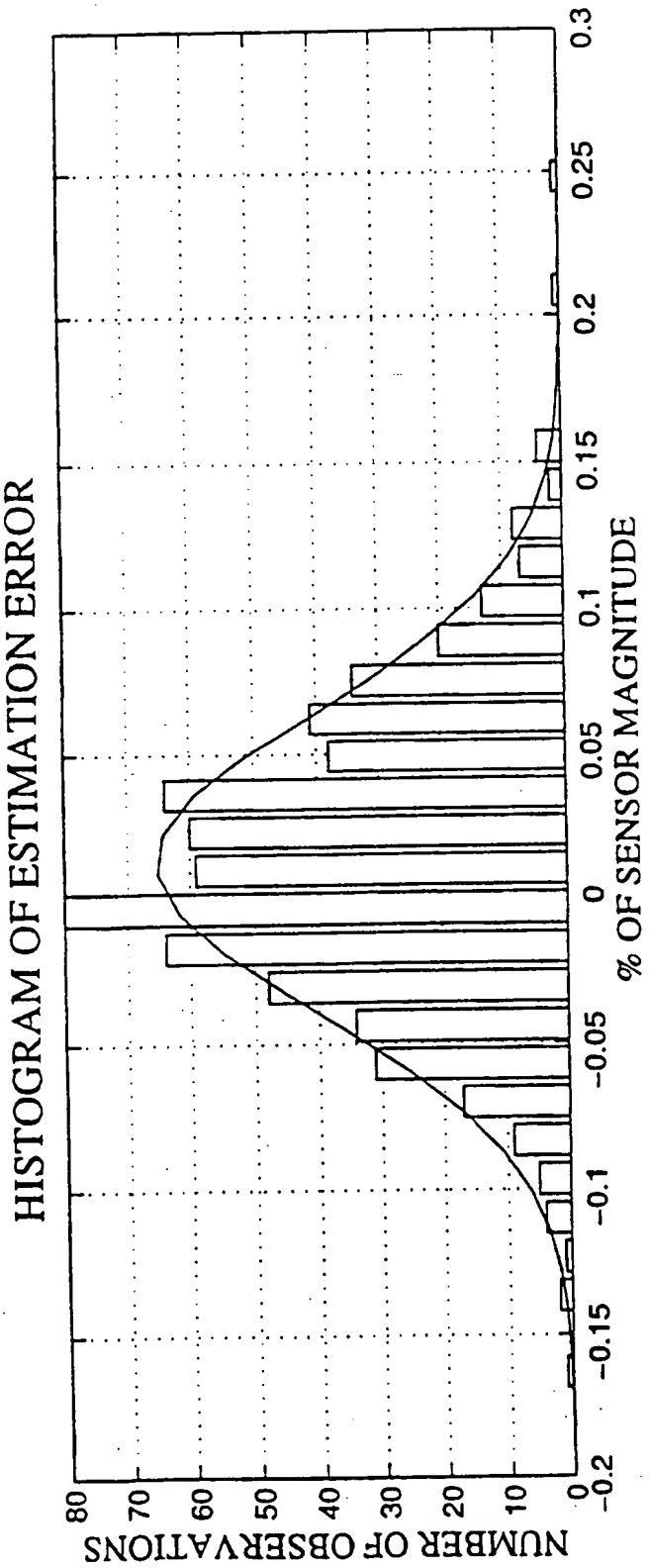

HISTOGRAM OF ESTIMATION ERROR
NUMBER OF OBSERVATIONS
% OF SENSOR MAGNITUDE
80
70
60
50
40
30
20
10
0
-0.2
-0.15
-0.1
-0.05
0
0.05
0.1
0.15
0.2
0.25
0.3

FIG. 7C

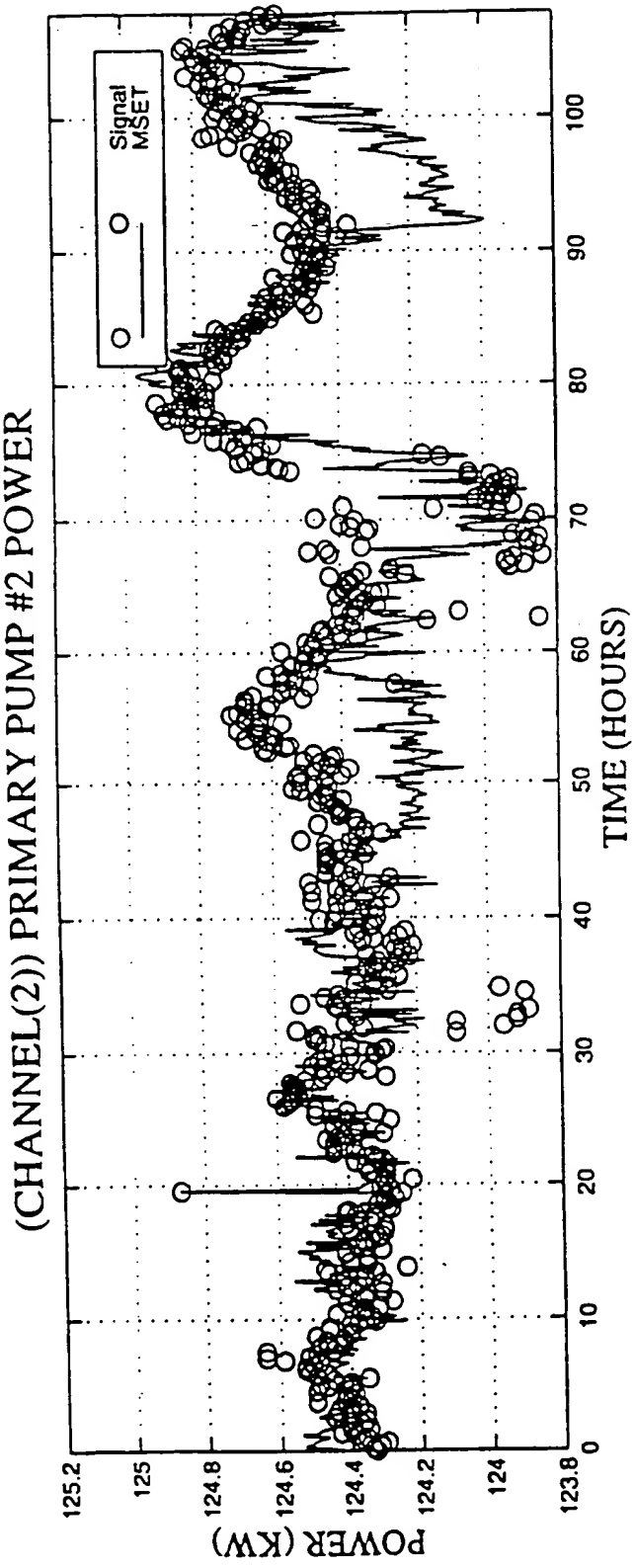

(CHANNEL(2)) PRIMARY PUMP #2 POWER
Signal
MSET
POWER (KW)
125.2
125
124.8
124.6
124.4
124.2
124
123.8
0
10
20
30
40
50
60
70
80
90
100
TIME (HOURS)

FIG. 8A

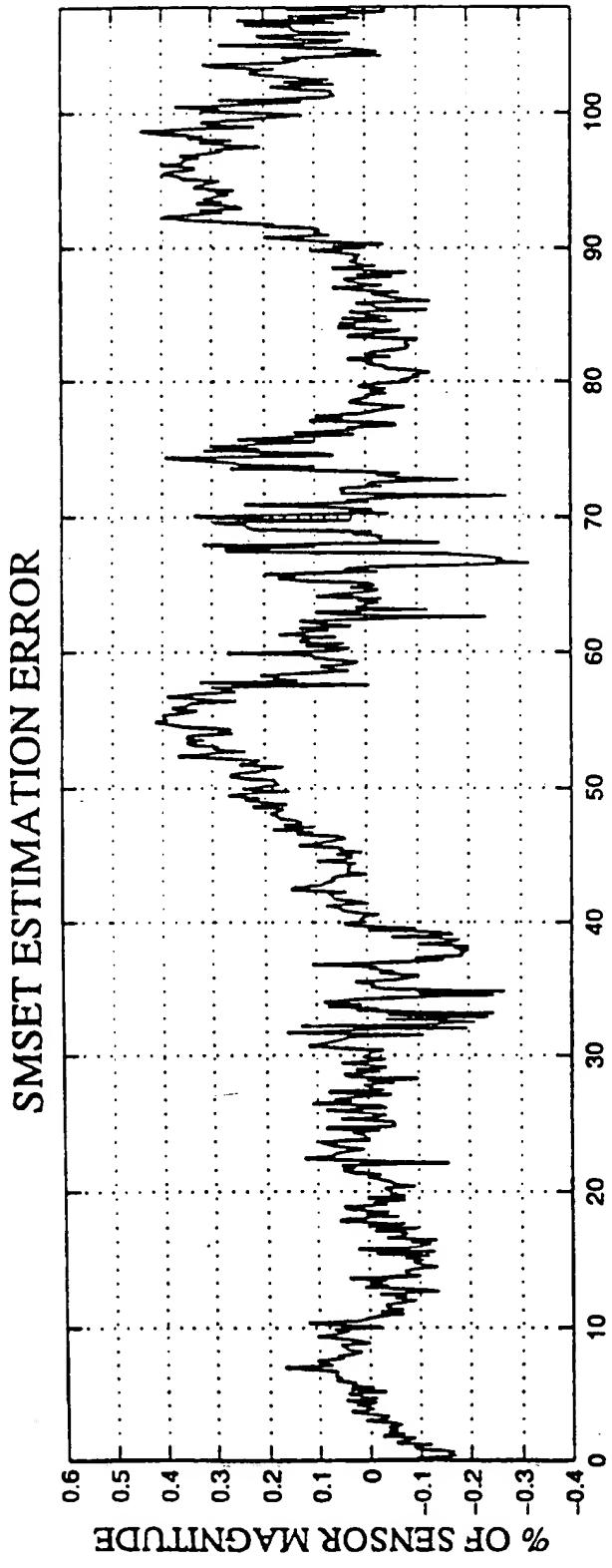

SMSET ESTIMATION ERROR
% OF SENSOR MAGNITUDE
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
0
10
20
30
40
50
60
70
80
90
100
TIME (HOURS)

FIG. 8B

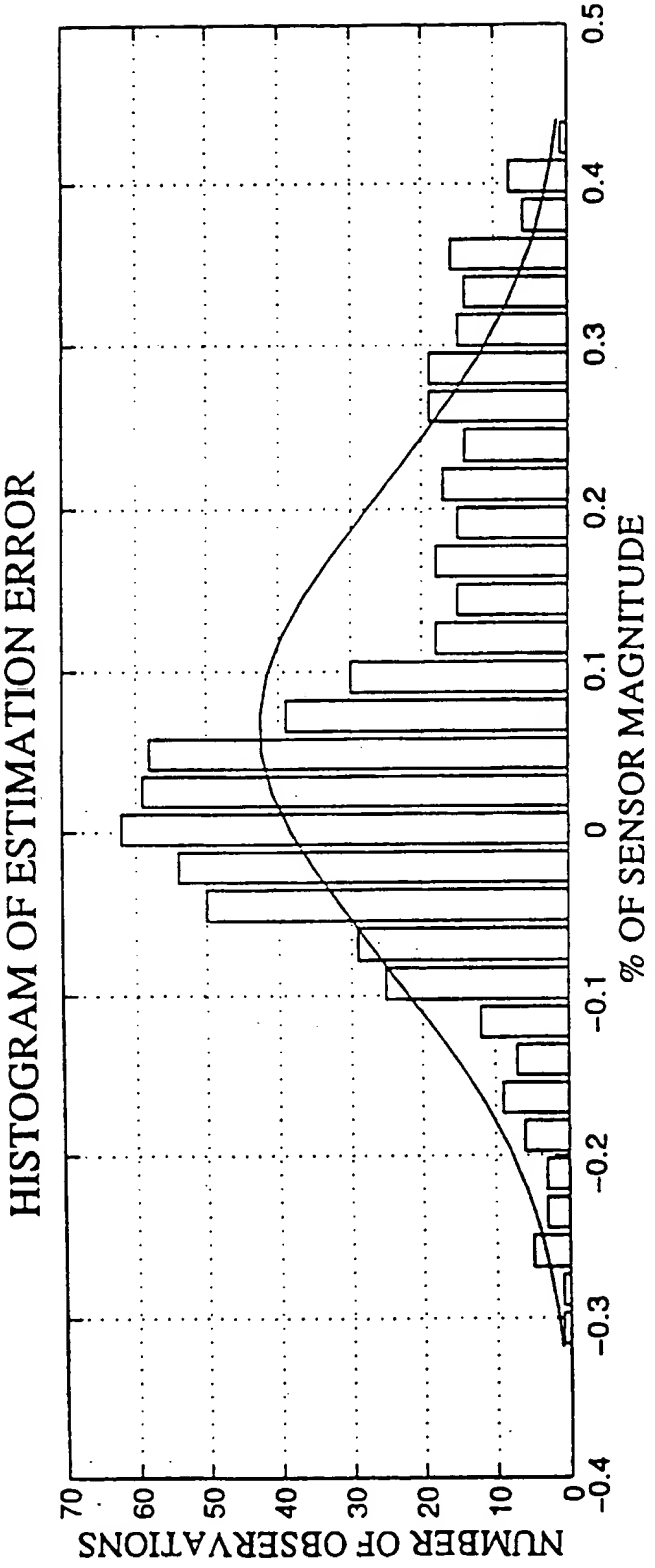

HISTOGRAM OF ESTIMATION ERROR
NUMBER OF OBSERVATIONS
70
60
50
40
30
20
10
0
-0.4
-0.3
-0.2
-0.1
0
0.1
0.2
0.3
0.4
0.5
% OF SENSOR MAGNITUDE

FIG. 8C

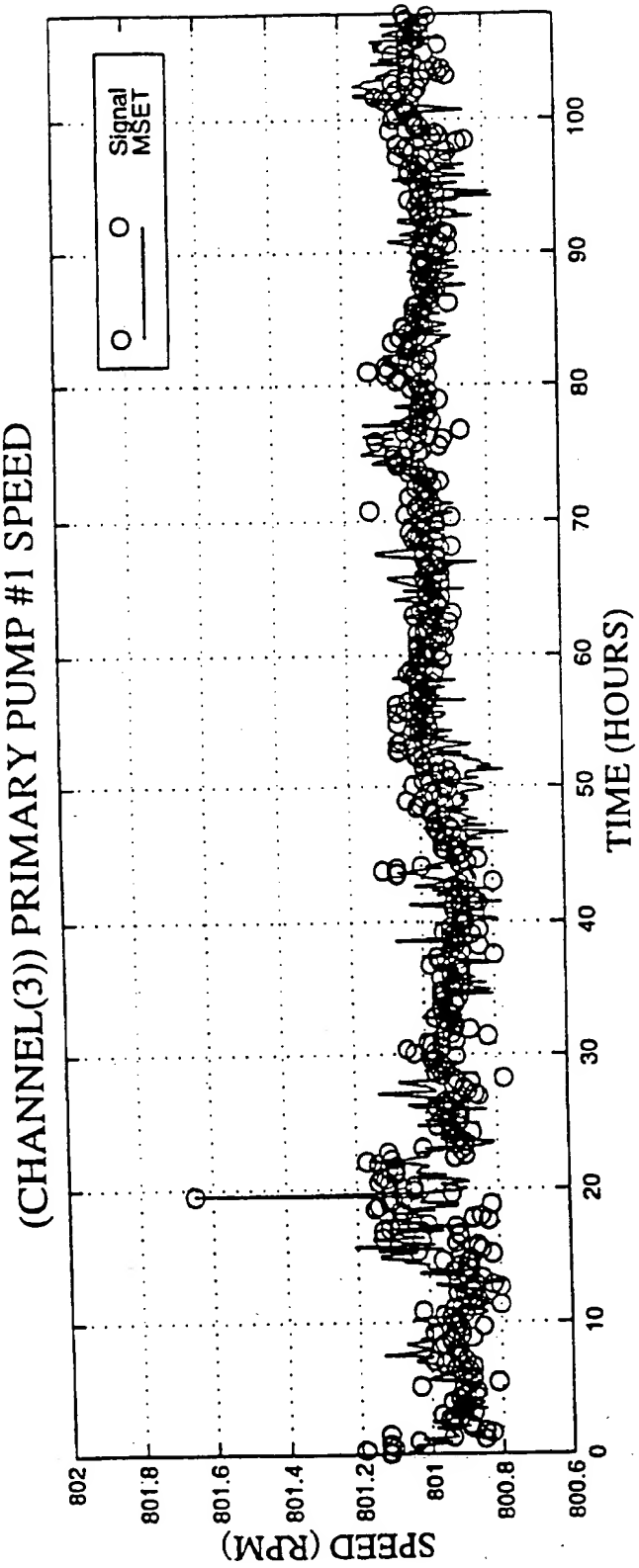

(CHANNEL(3)) PRIMARY PUMP #1 SPEED
Signal
MSET
SPEED (RPM)
802
801.8
801.6
801.4
801.2
801
800.8
800.6
TIME (HOURS)
0
10
20
30
40
50
60
70
80
90
100

FIG. 9A

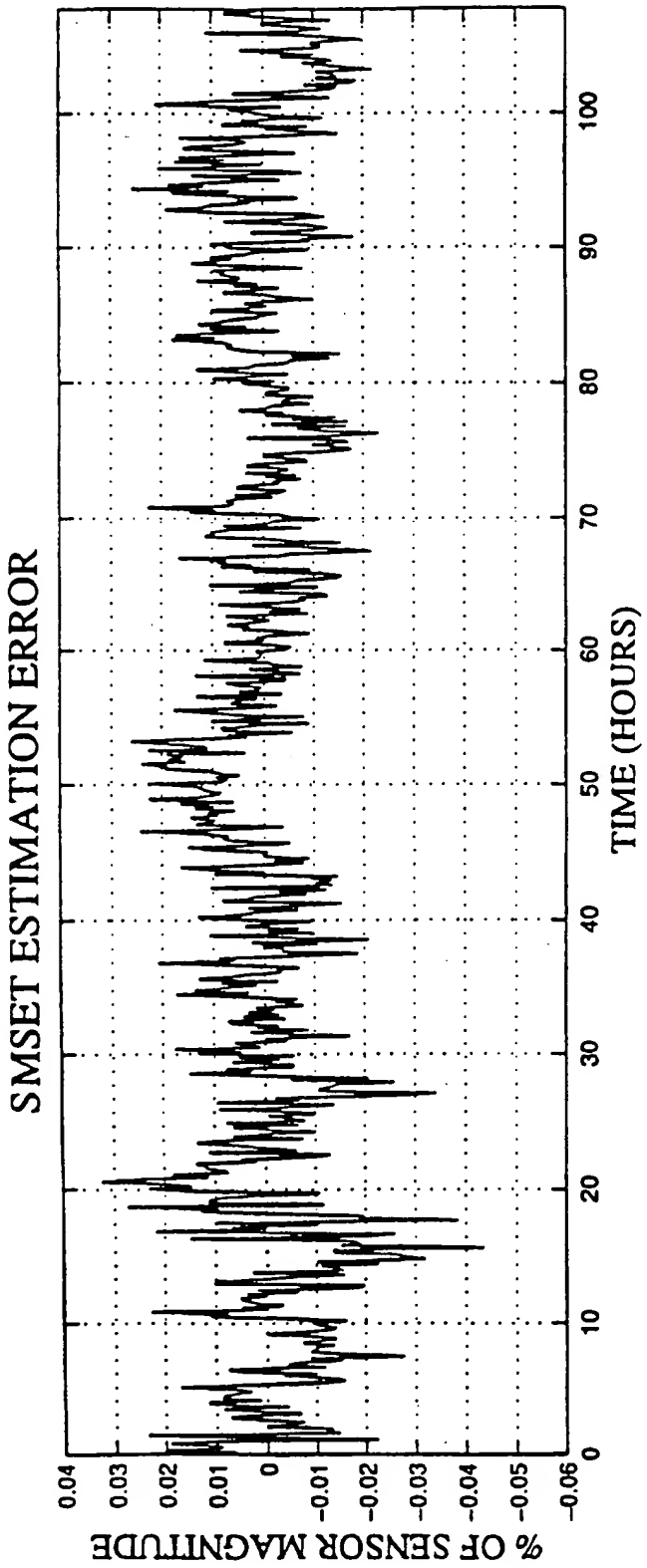

SMSET ESTIMATION ERROR
% OF SENSOR MAGNITUDE
0.04
0.03
0.02
0.01
0
-0.01
-0.02
-0.03
-0.04
-0.05
-0.06
0
10
20
30
40
50
60
70
80
90
100
TIME (HOURS)

FIG. 9B

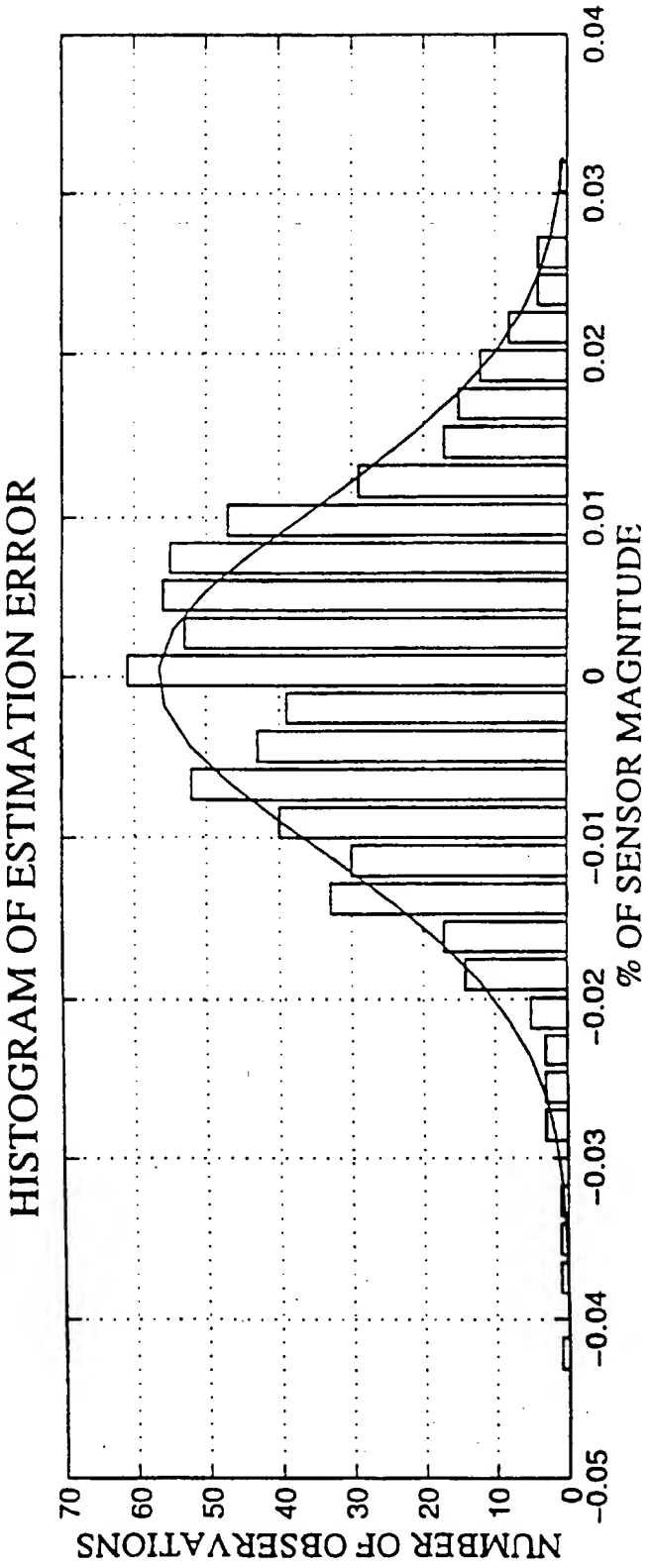

HISTOGRAM OF ESTIMATION ERROR
NUMBER OF OBSERVATIONS
0
10
20
30
40
50
60
70
-0.05
-0.04
-0.03
-0.02
-0.01
0
0.01
0.02
0.03
0.04
% OF SENSOR MAGNITUDE

FIG. 9C

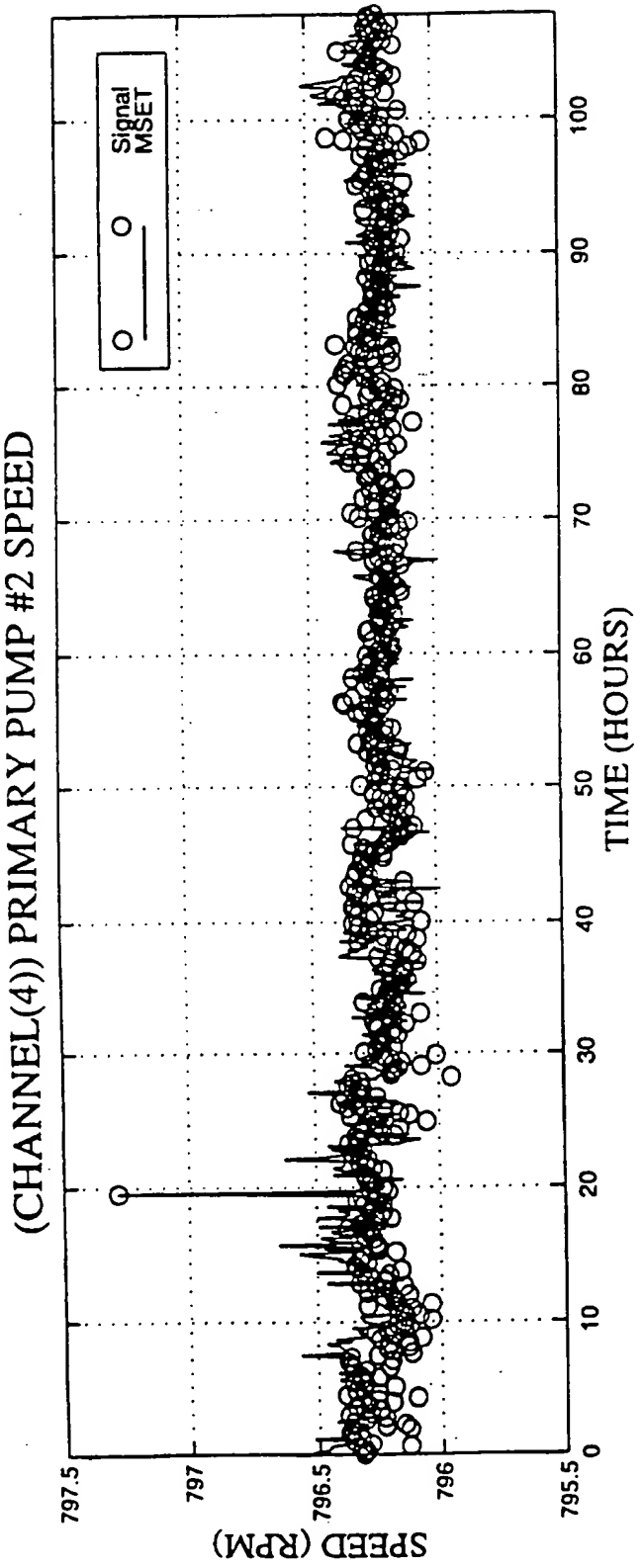

(CHANNEL(4)) PRIMARY PUMP #2 SPEED
Signal
MSET
797.5
797
796.5
796
795.5
SPEED (RPM)
0
10
20
30
40
50
60
70
80
90
100
TIME (HOURS)

FIG. 10A

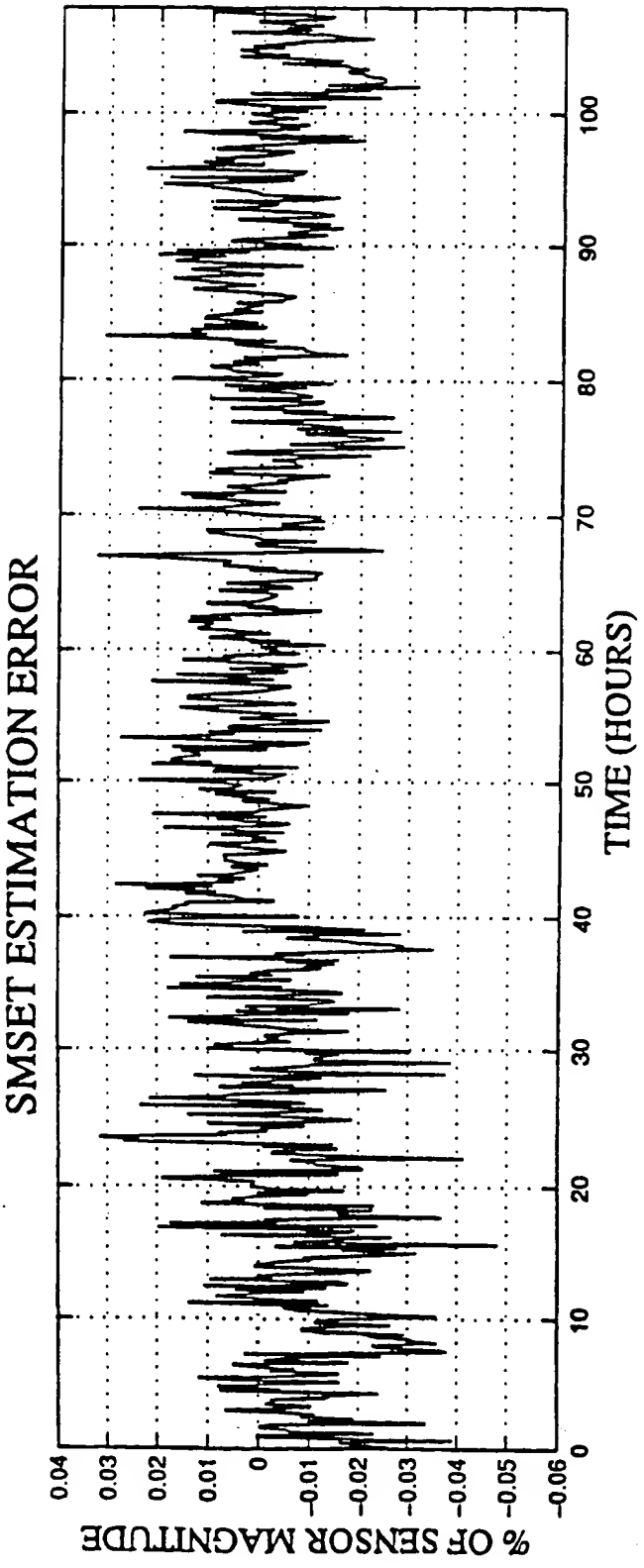

SMSET ESTIMATION ERROR
% OF SENSOR MAGNITUDE
0.04
0.03
0.02
0.01
0
-0.01
-0.02
-0.03
-0.04
-0.05
-0.06
0
10
20
30
40
50
60
70
80
90
100
TIME (HOURS)

FIG. 10B

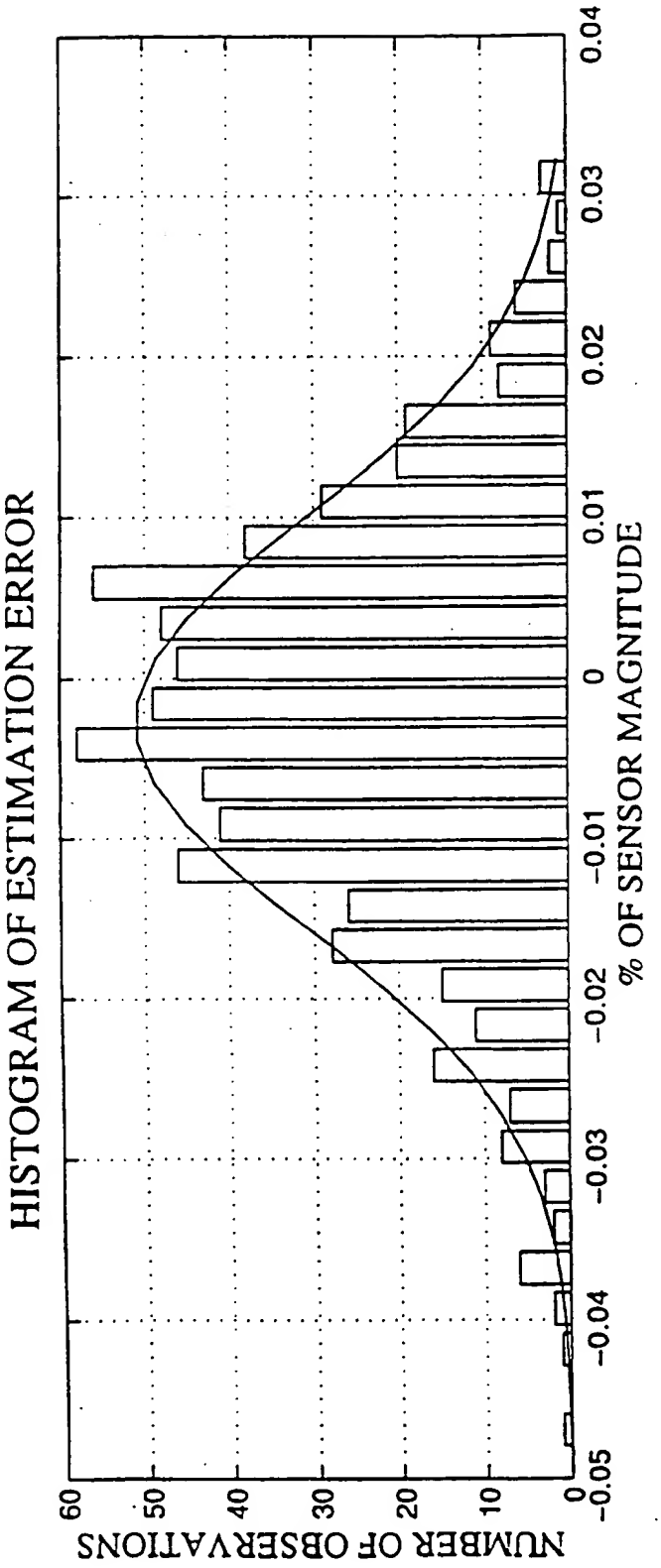

HISTOGRAM OF ESTIMATION ERROR
NUMBER OF OBSERVATIONS
60
50
40
30
20
10
0
-0.05
-0.04
-0.03
-0.02
-0.01
0
0.01
0.02
0.03
0.04
% OF SENSOR MAGNITUDE

FIG. 10C

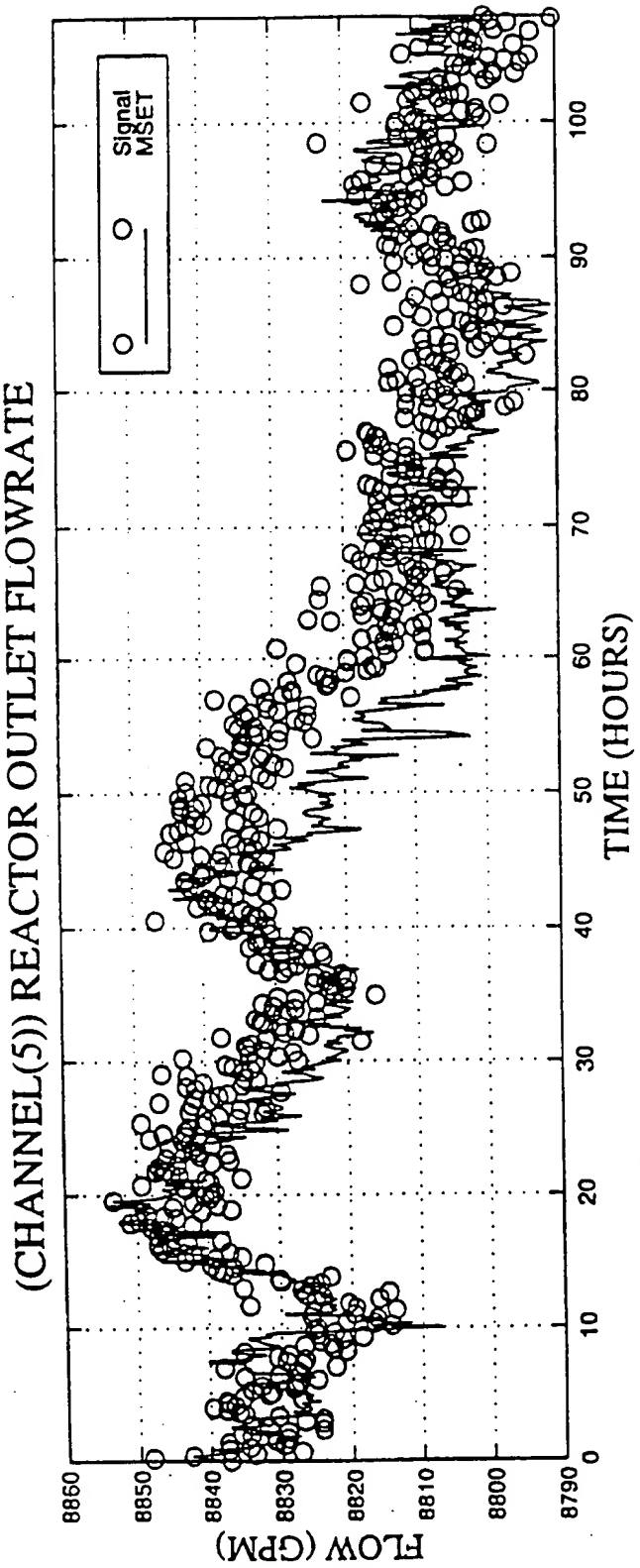

(CHANNEL(5)) REACTOR OUTLET FLOWRATE
Signal
MSET
FLOW (GPM)
8860
8850
8840
8830
8820
8810
8800
8790
TIME (HOURS)
0
10
20
30
40
50
60
70
80
90
100

FIG. 11A

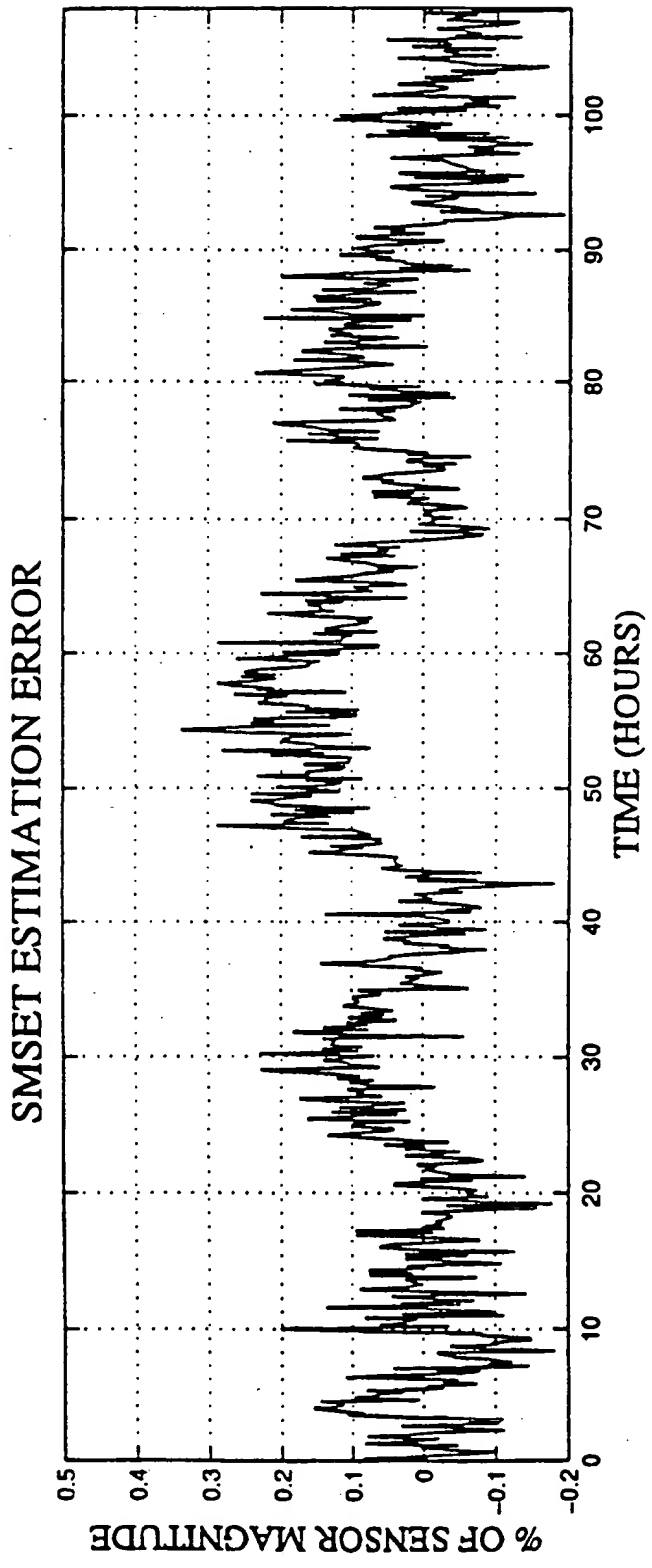

SMSET ESTIMATION ERROR
% OF SENSOR MAGNITUDE
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
TIME (HOURS)
0
10
20
30
40
50
60
70
80
90
100

FIG. 11B

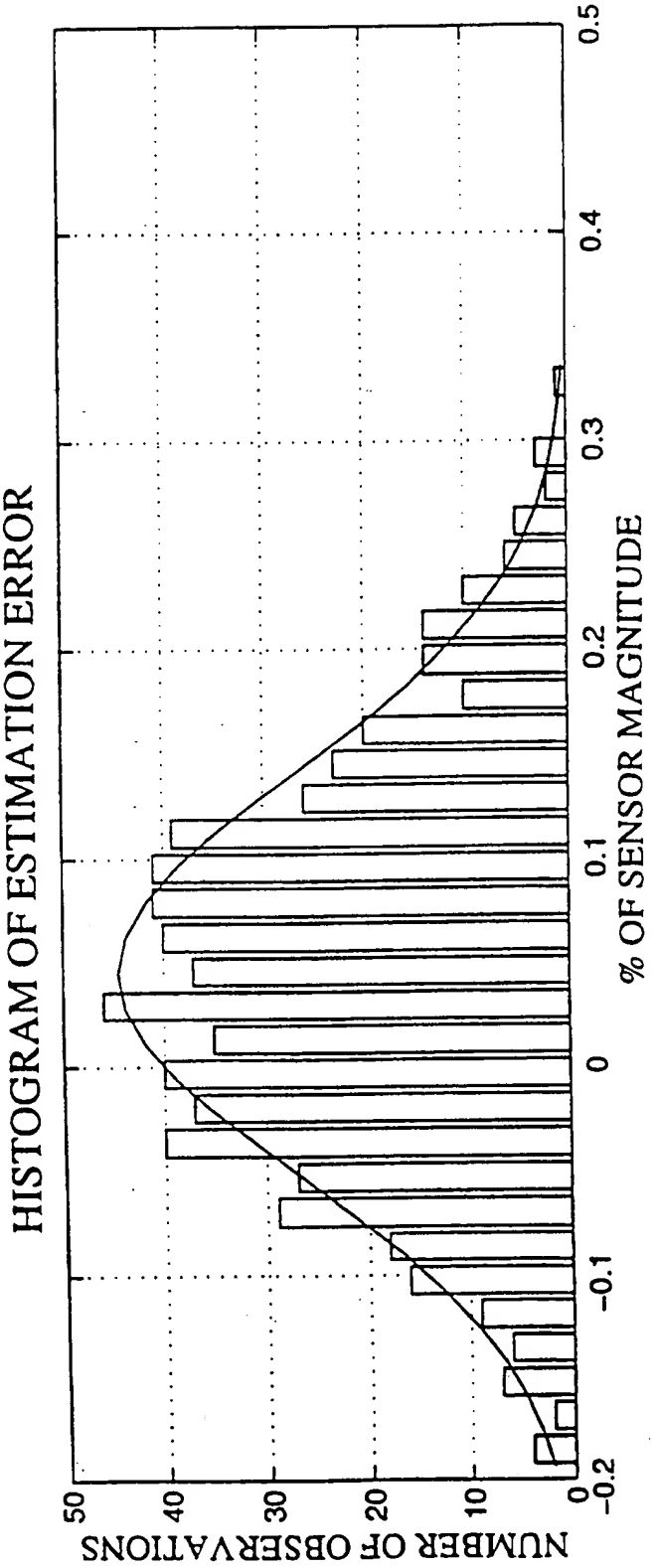

HISTOGRAM OF ESTIMATION ERROR
NUMBER OF OBSERVATIONS
% OF SENSOR MAGNITUDE
50
40
30
20
10
0
-0.2
-0.1
0
0.1
0.2
0.3
0.4
0.5

FIG. 11C

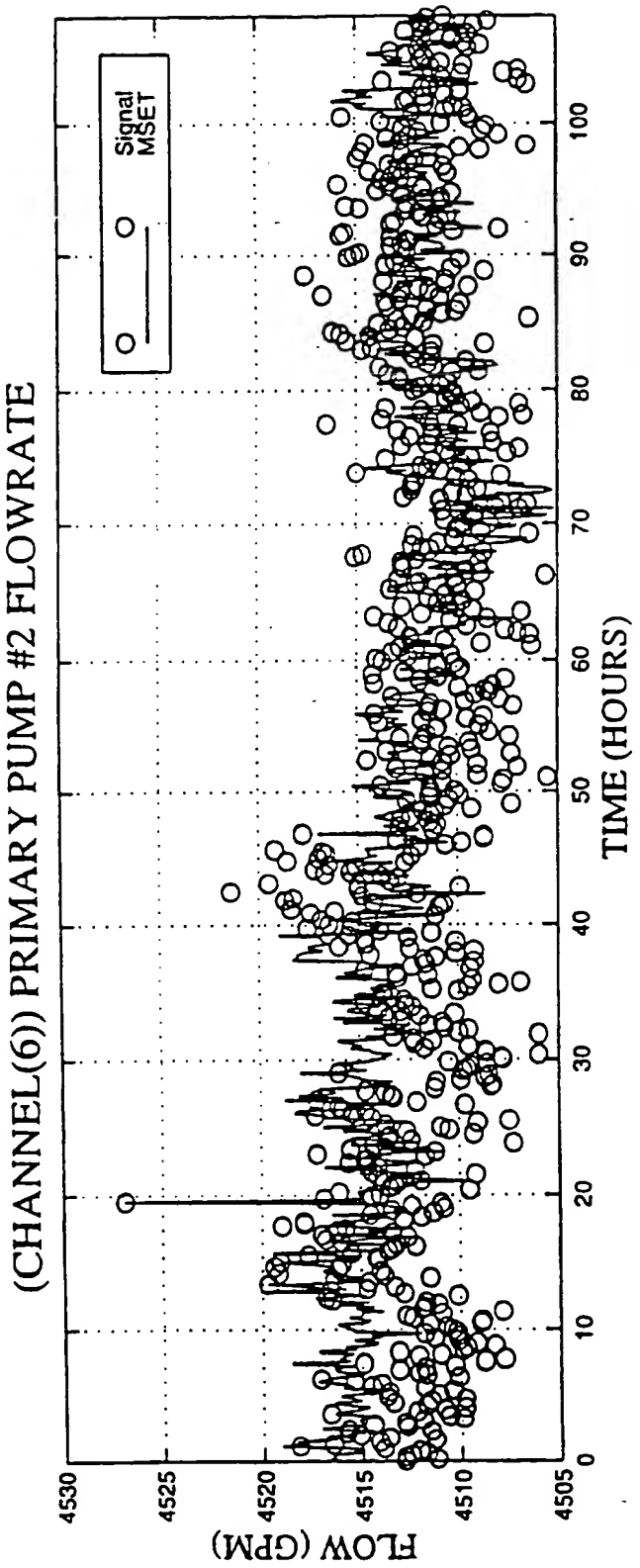

(CHANNEL(6)) PRIMARY PUMP #2 FLOWRATE
Signal
MSET
FLOW (GPM)
4530
4525
4520
4515
4510
4505
0
10
20
30
40
50
60
70
80
90
100
TIME (HOURS)

FIG. 12A

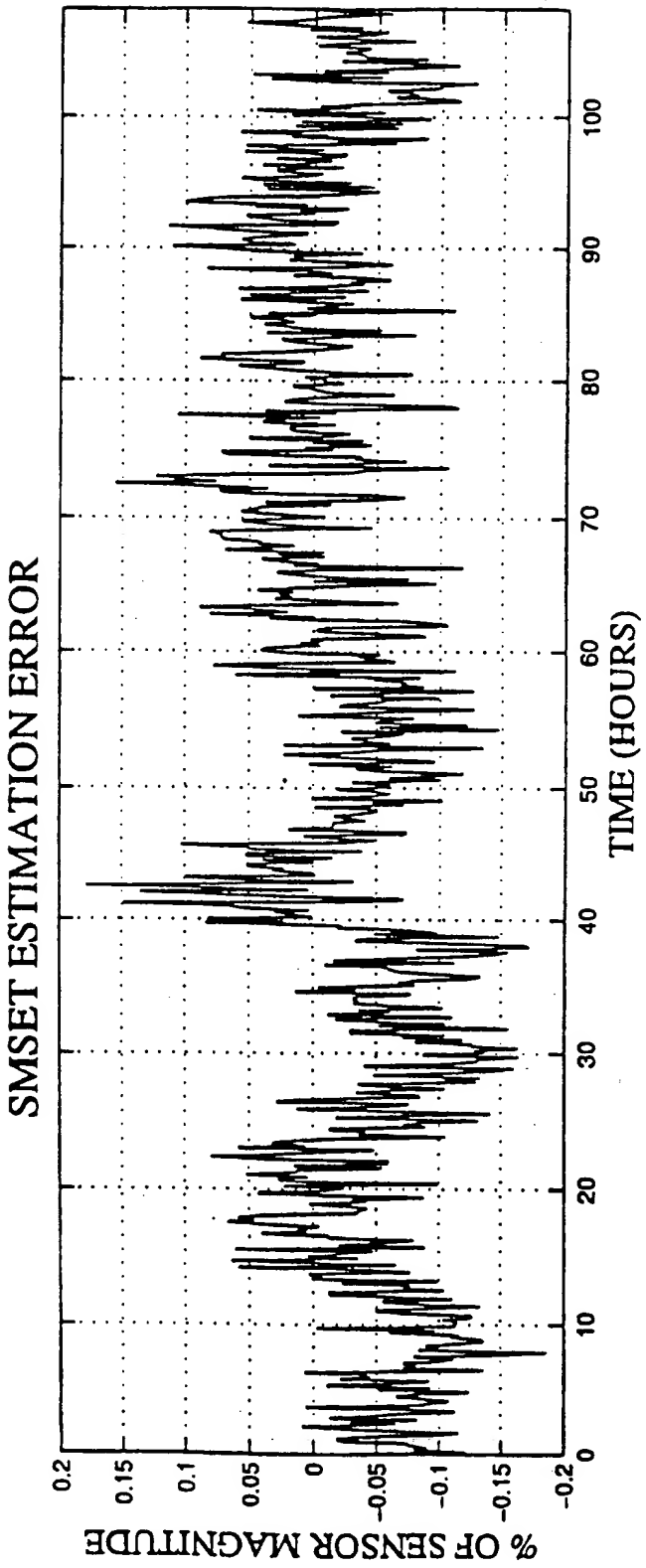

SMSET ESTIMATION ERROR
% OF SENSOR MAGNITUDE
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
0
10
20
30
40
50
60
70
80
90
100
TIME (HOURS)

FIG. 12B

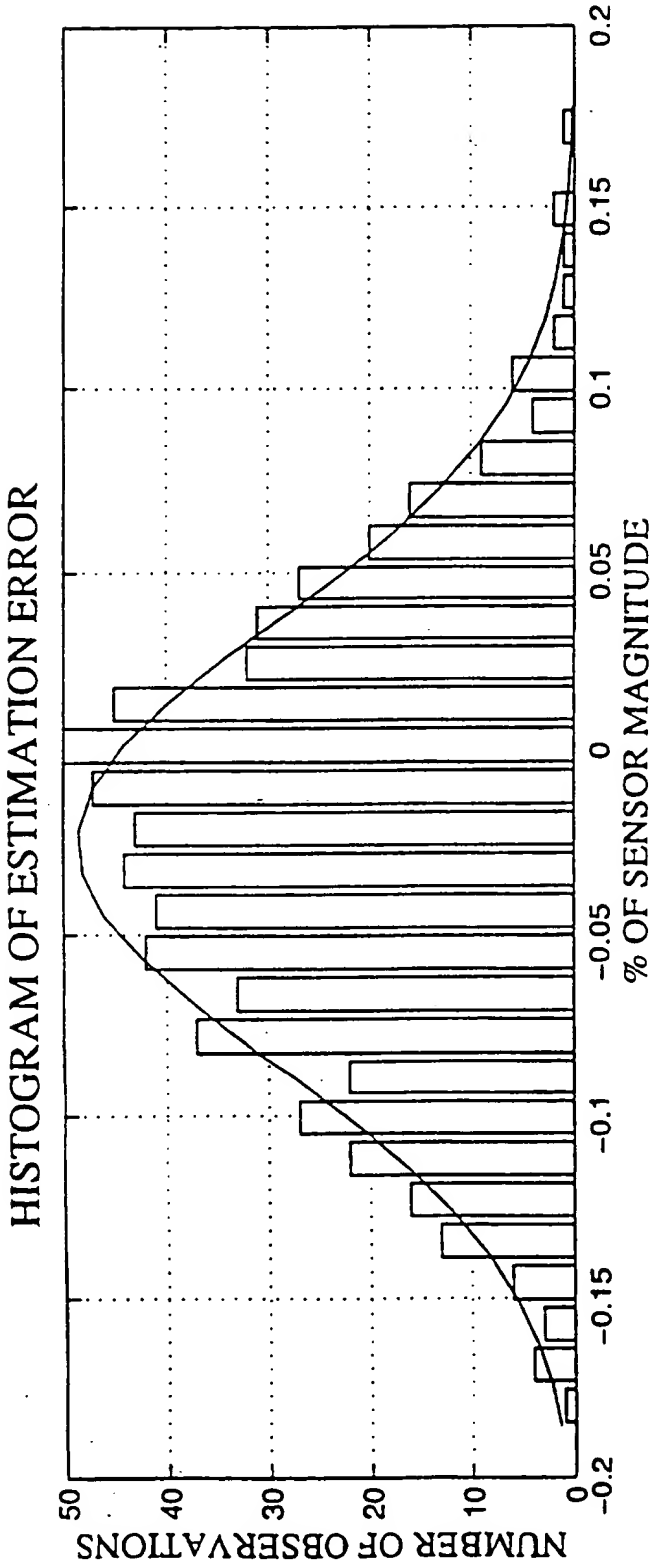

HISTOGRAM OF ESTIMATION ERROR
NUMBER OF OBSERVATIONS
% OF SENSOR MAGNITUDE
50
40
30
20
10
0
-0.2
-0.15
-0.1
-0.05
0
0.05
0.1
0.15
0.2

FIG. 12C

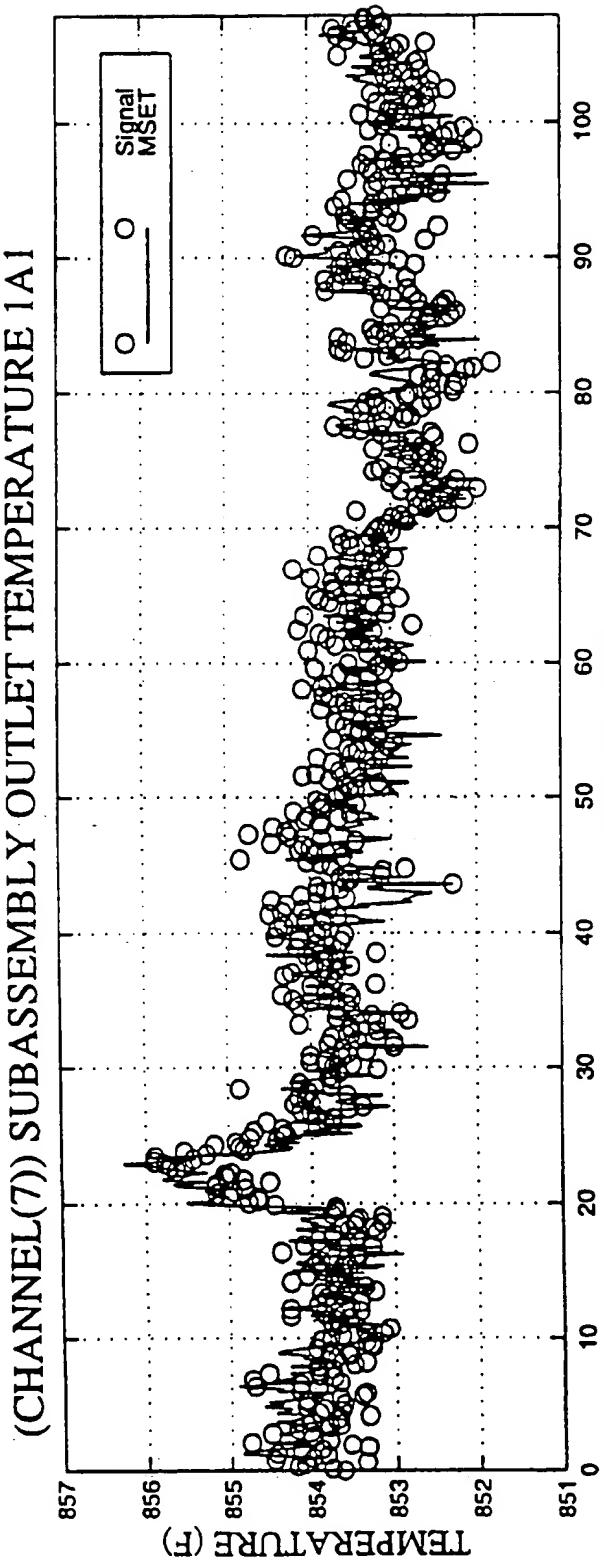

(CHANNEL(7)) SUBASSEMBLY OUTLET TEMPERATURE 1A1
Signal
MSET
TEMPERATURE (F)
857
856
855
854
853
852
851
0
10
20
30
40
50
60
70
80
90
100
TIME (HOURS)

FIG. 13A

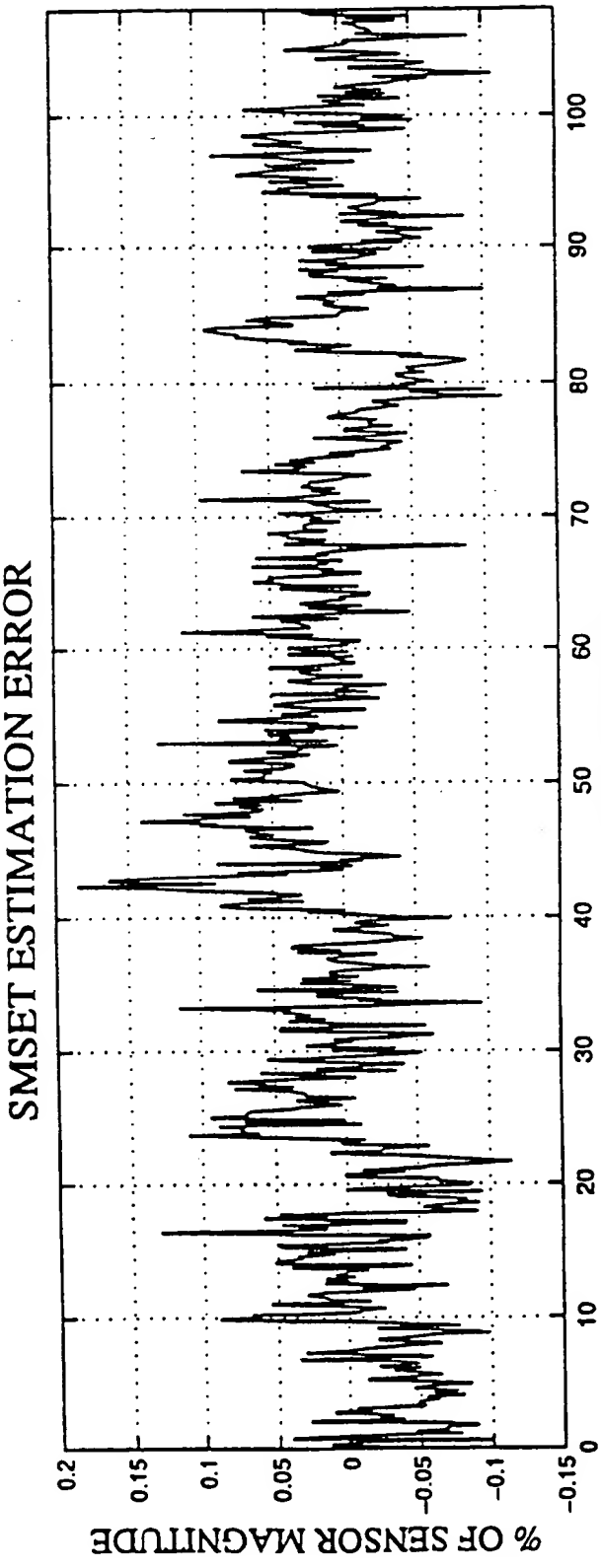

SMSET ESTIMATION ERROR
% OF SENSOR MAGNITUDE
TIME (HOURS)
0.2
0.15
0.1
0.05
0
−0.05
−0.1
−0.15
0
10
20
30
40
50
60
70
80
90
100

FIG. 13B

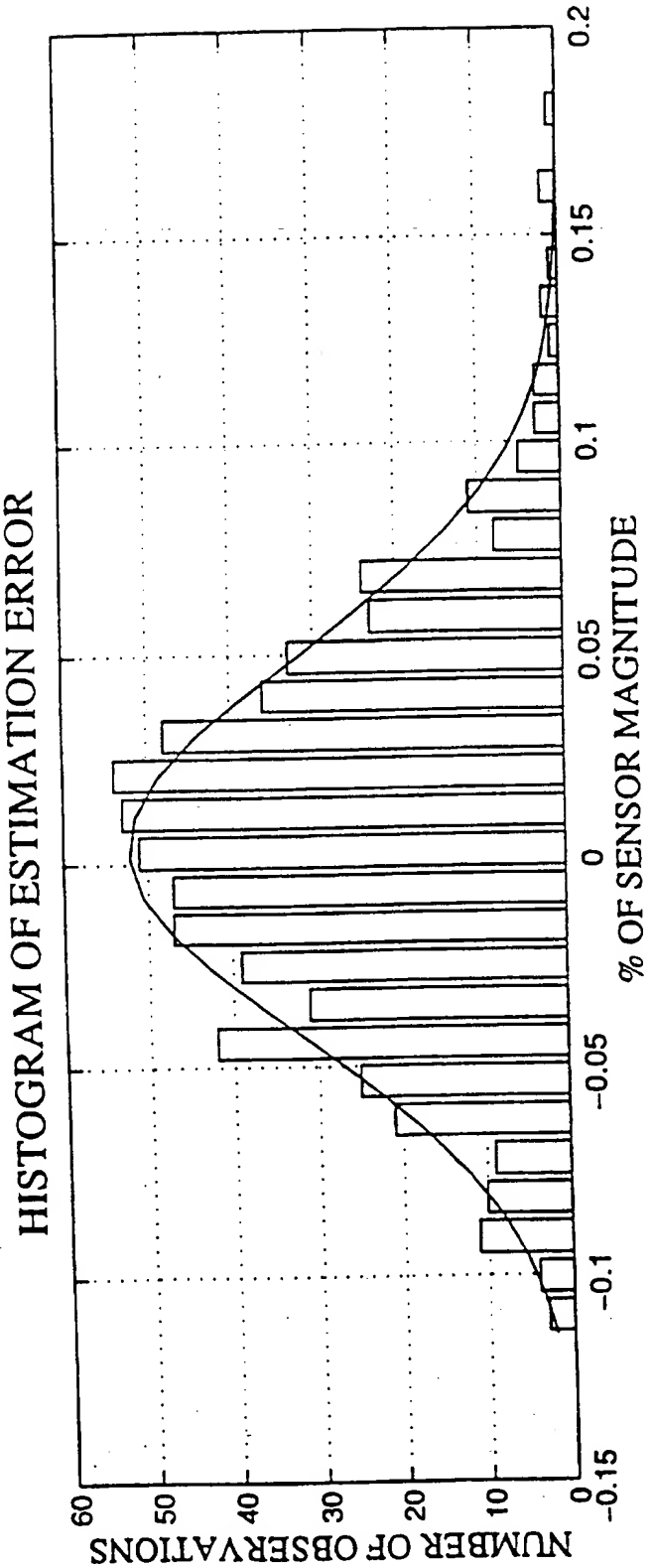

HISTOGRAM OF ESTIMATION ERROR
NUMBER OF OBSERVATIONS
% OF SENSOR MAGNITUDE
60
50
40
30
20
10
0
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15

FIG. 13C

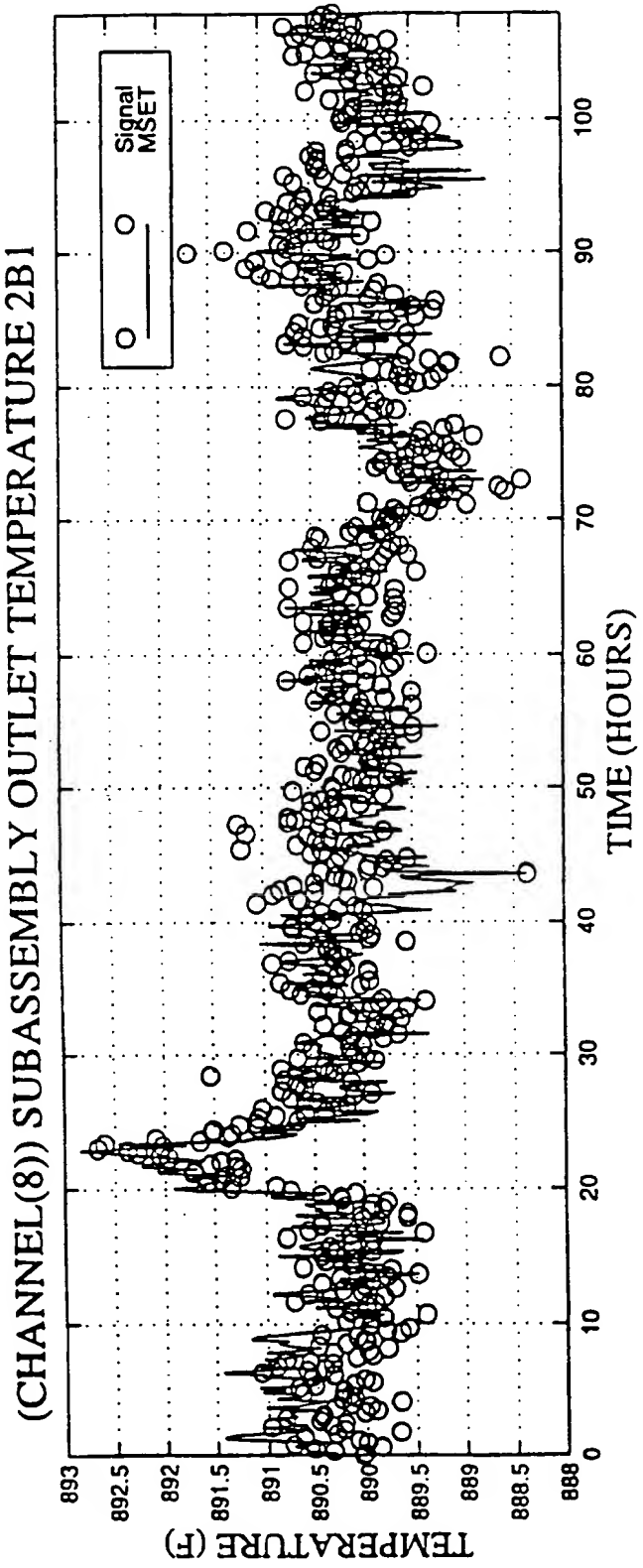

(CHANNEL(8)) SUBASSEMBLY OUTLET TEMPERATURE 2B1
Signal
MSET
TEMPERATURE (F)
TIME (HOURS)
893
892.5
892
891.5
891
890.5
890
889.5
889
888.5
888
0
10
20
30
40
50
60
70
80
90
100

FIG. 14A

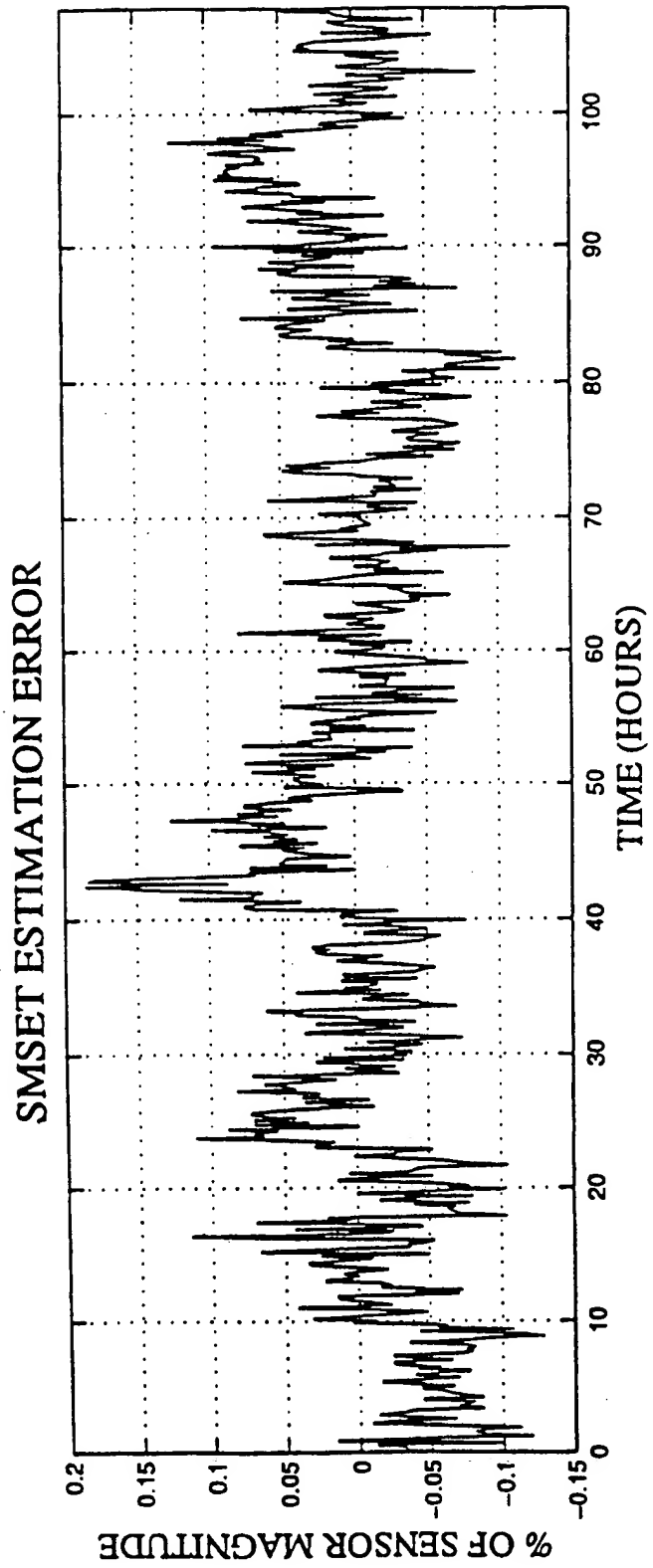

SMSET ESTIMATION ERROR
% OF SENSOR MAGNITUDE
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
0
10
20
30
40
50
60
70
80
90
100
TIME (HOURS)

FIG. 14B

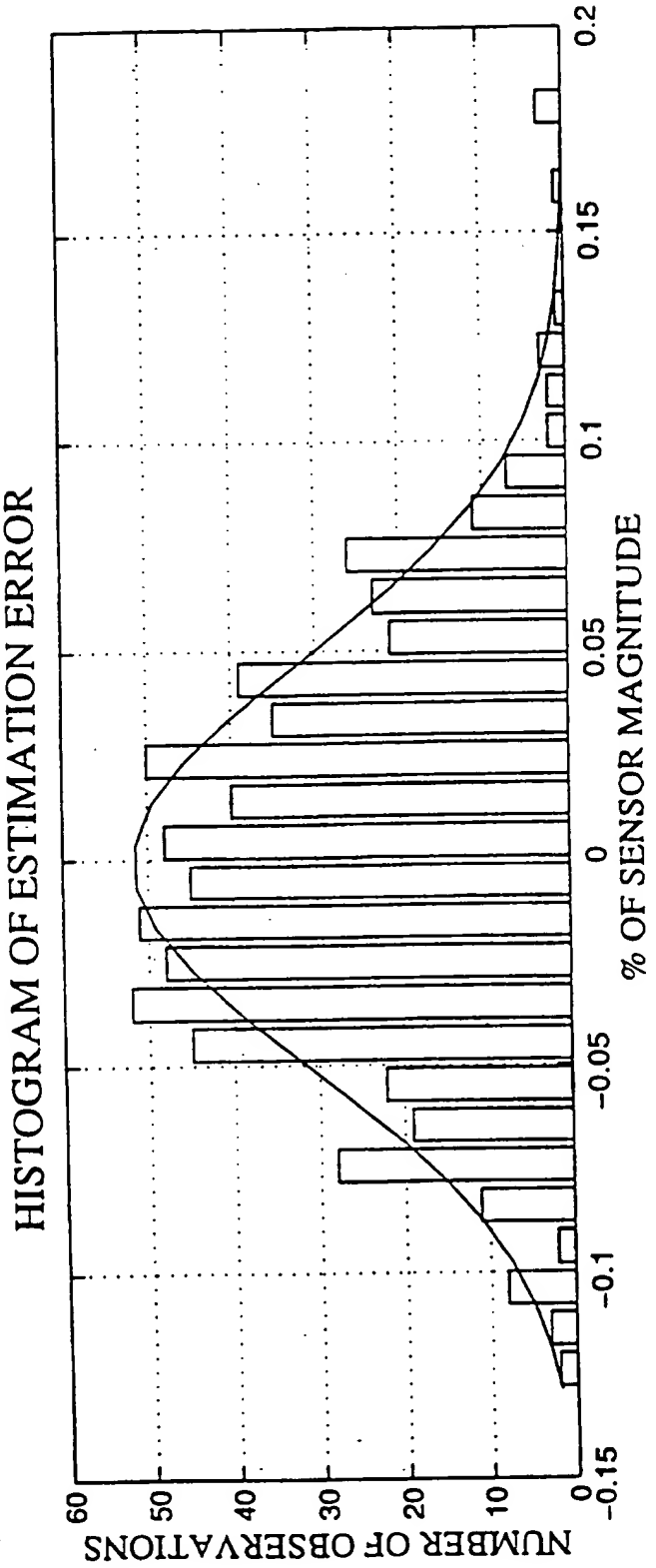

HISTOGRAM OF ESTIMATION ERROR
NUMBER OF OBSERVATIONS
% OF SENSOR MAGNITUDE
60
50
40
30
20
10
0
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15

FIG. 14C

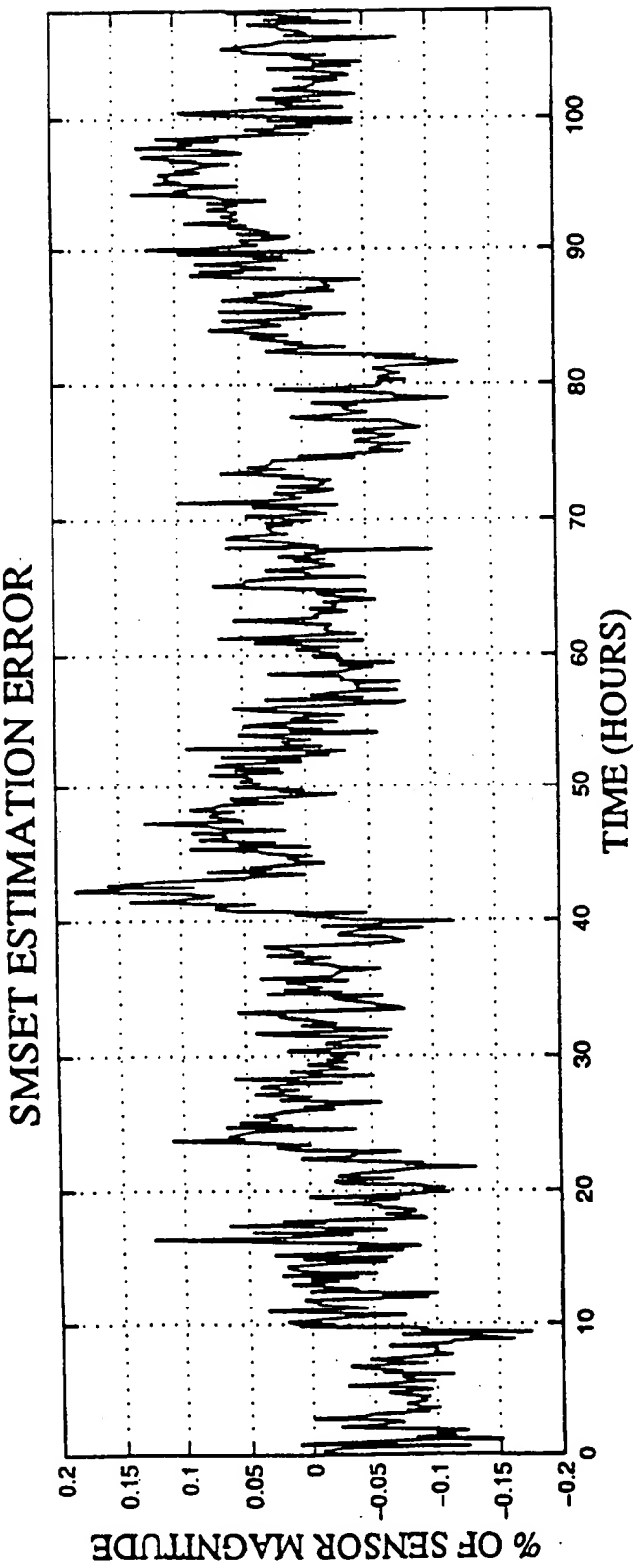

SMSET ESTIMATION ERROR
% OF SENSOR MAGNITUDE
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
0
10
20
30
40
50
60
70
80
90
100
TIME (HOURS)

FIG. 15B

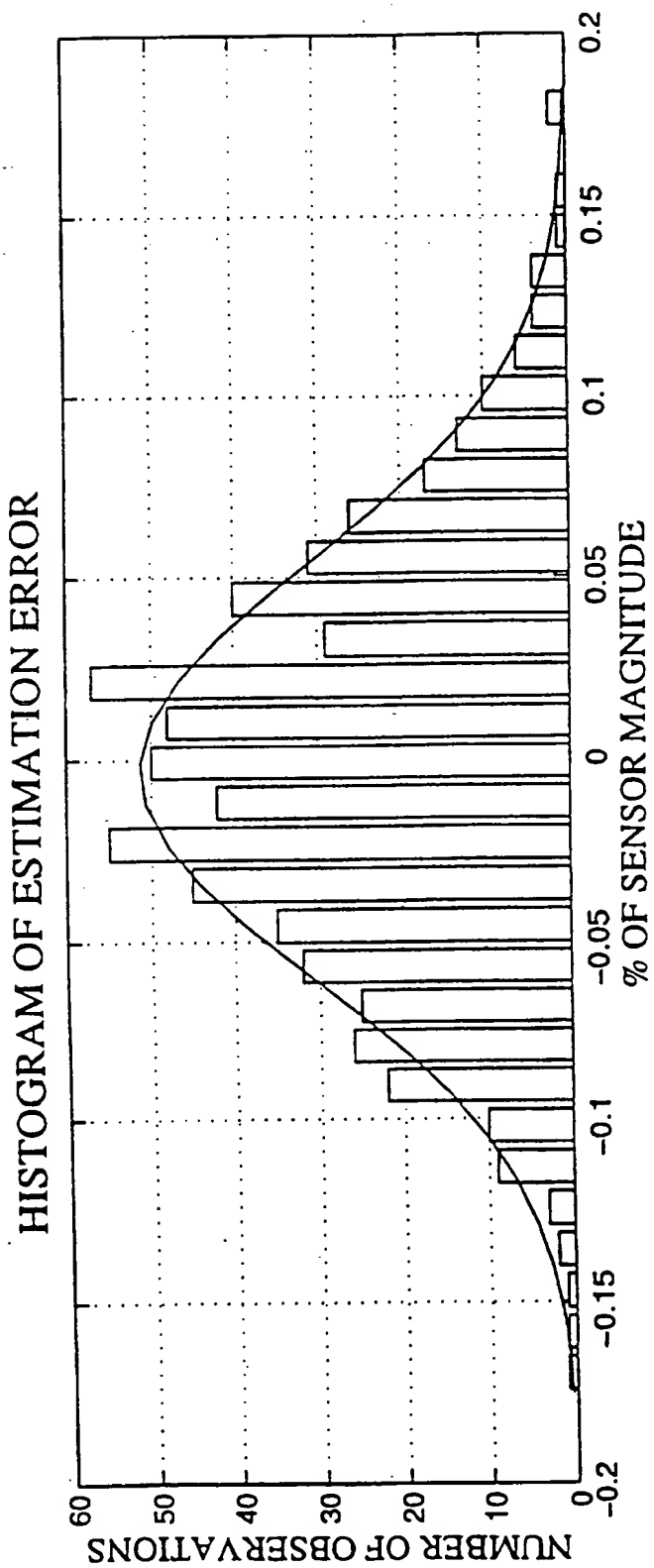

HISTOGRAM OF ESTIMATION ERROR
NUMBER OF OBSERVATIONS
60
50
40
30
20
10
0
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
% OF SENSOR MAGNITUDE

FIG. 15C

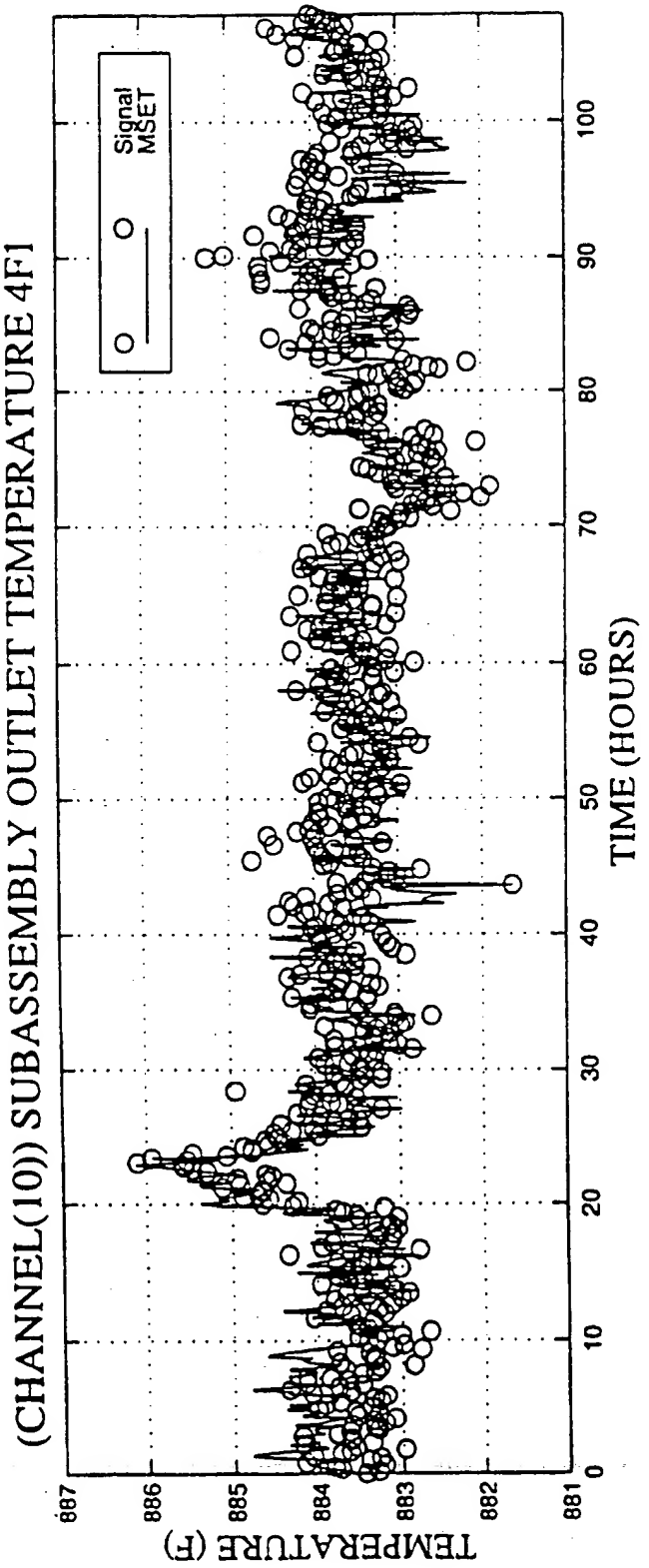

(CHANNEL(10)) SUBASSEMBLY OUTLET TEMPERATURE 4F1
Signal
MSET
TEMPERATURE (F)
887
886
885
884
883
882
881
TIME (HOURS)
0
10
20
30
40
50
60
70
80
90
100

FIG. 16A

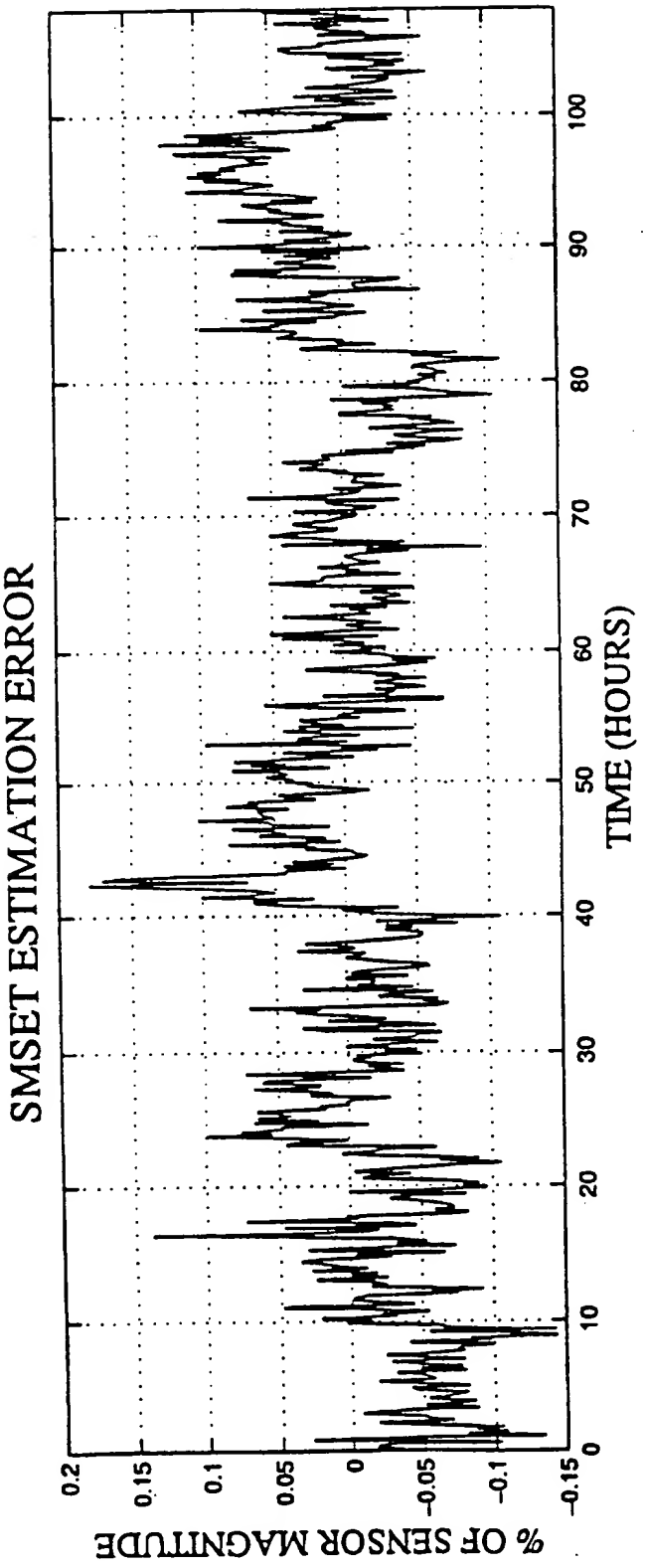

SMSET ESTIMATION ERROR
% OF SENSOR MAGNITUDE
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
0
10
20
30
40
50
60
70
80
90
100
TIME (HOURS)

FIG. 16B

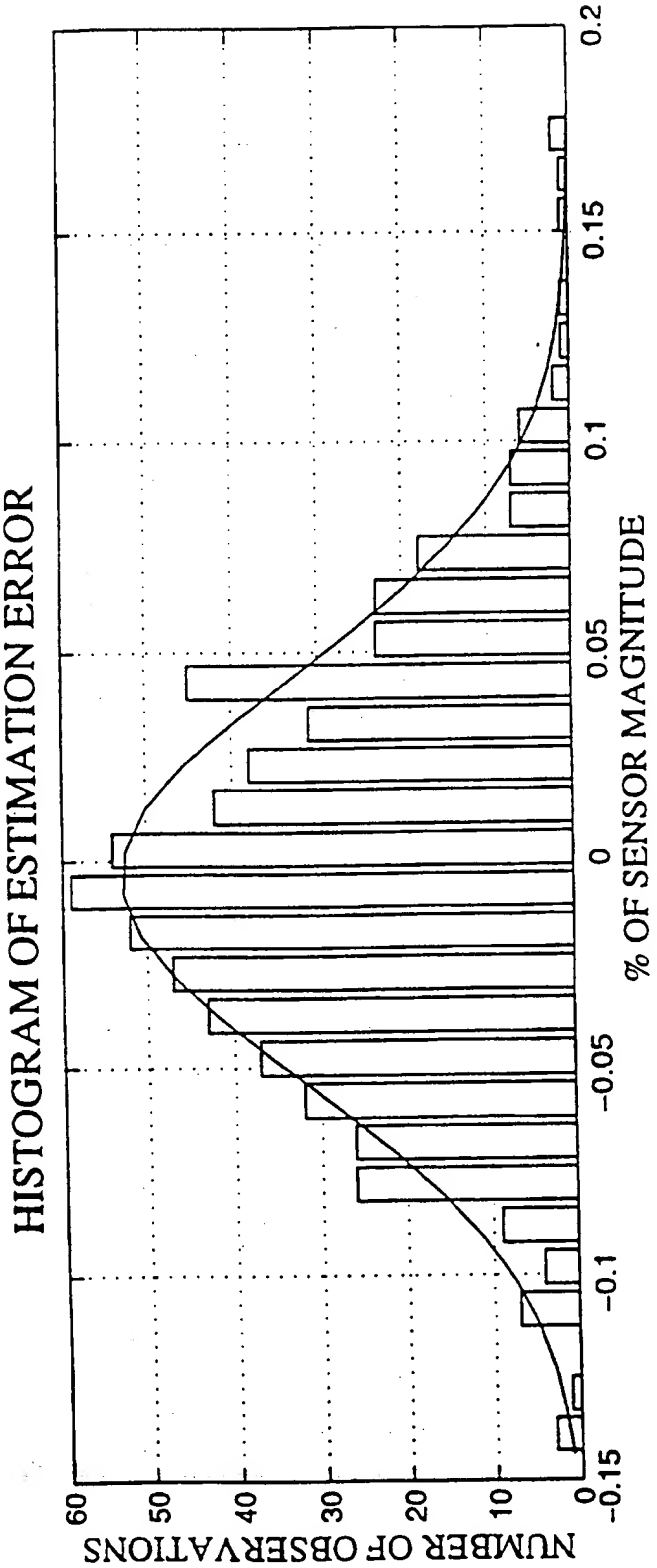

HISTOGRAM OF ESTIMATION ERROR
NUMBER OF OBSERVATIONS
60
50
40
30
20
10
0
-0.15
-0.1
-0.05
0
0.05
0.1
0.15
0.2
% OF SENSOR MAGNITUDE

FIG. 16C

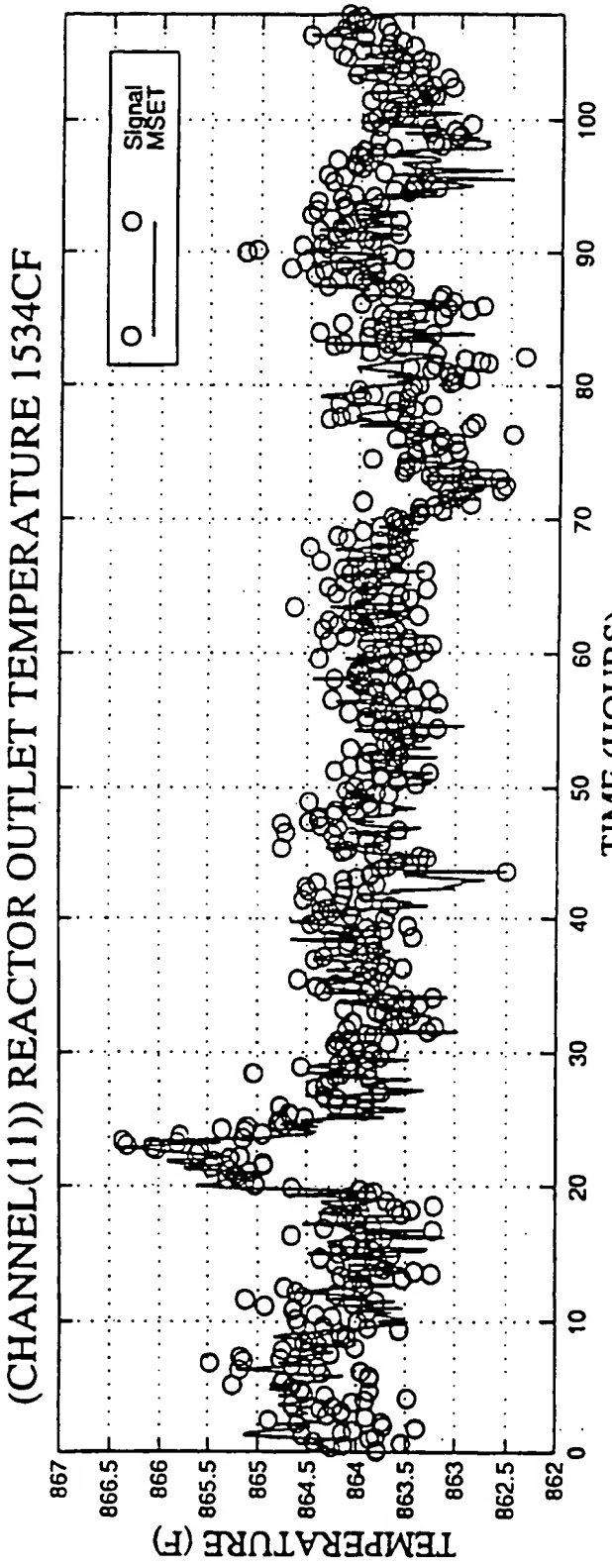

(CHANNEL(11)) REACTOR OUTLET TEMPERATURE 1534CF
Signal
MSET
TEMPERATURE (F)
867
866.5
866
865.5
865
864.5
864
863.5
863
862.5
862
TIME (HOURS)
0
10
20
30
40
50
60
70
80
90
100

FIG. 17A

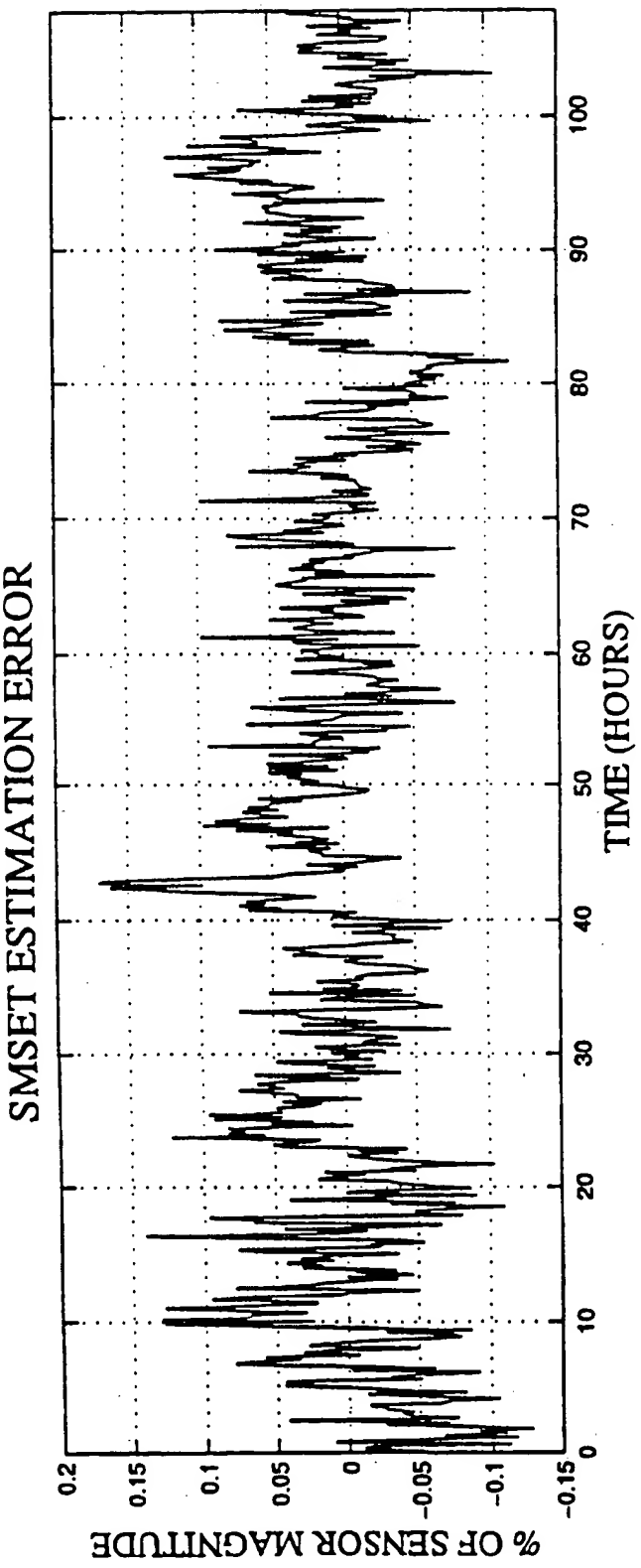

SMSET ESTIMATION ERROR
% OF SENSOR MAGNITUDE
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
TIME (HOURS)
0
10
20
30
40
50
60
70
80
90
100

FIG. 17B

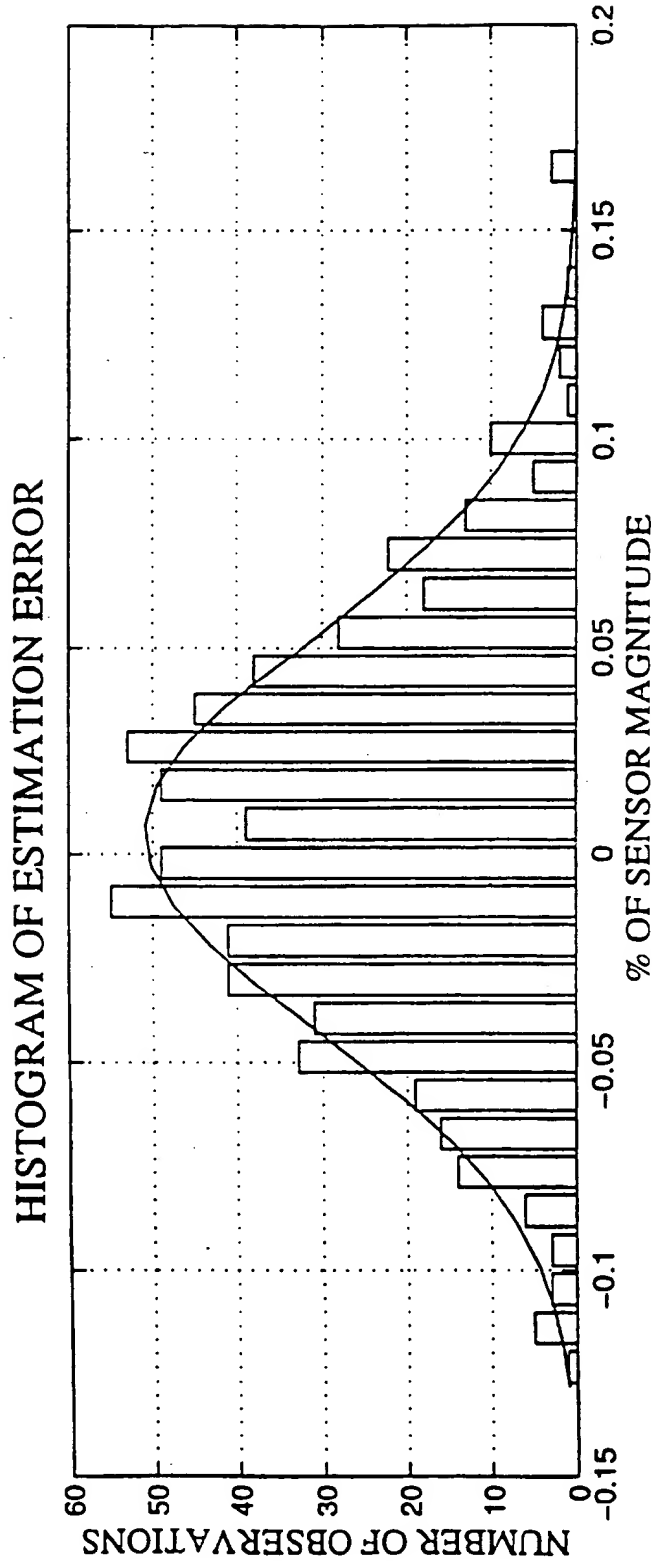

HISTOGRAM OF ESTIMATION ERROR
NUMBER OF OBSERVATIONS
0
10
20
30
40
50
60
% OF SENSOR MAGNITUDE
-0.15
-0.1
-0.05
0
0.05
0.1
0.15
0.2

FIG. 17C

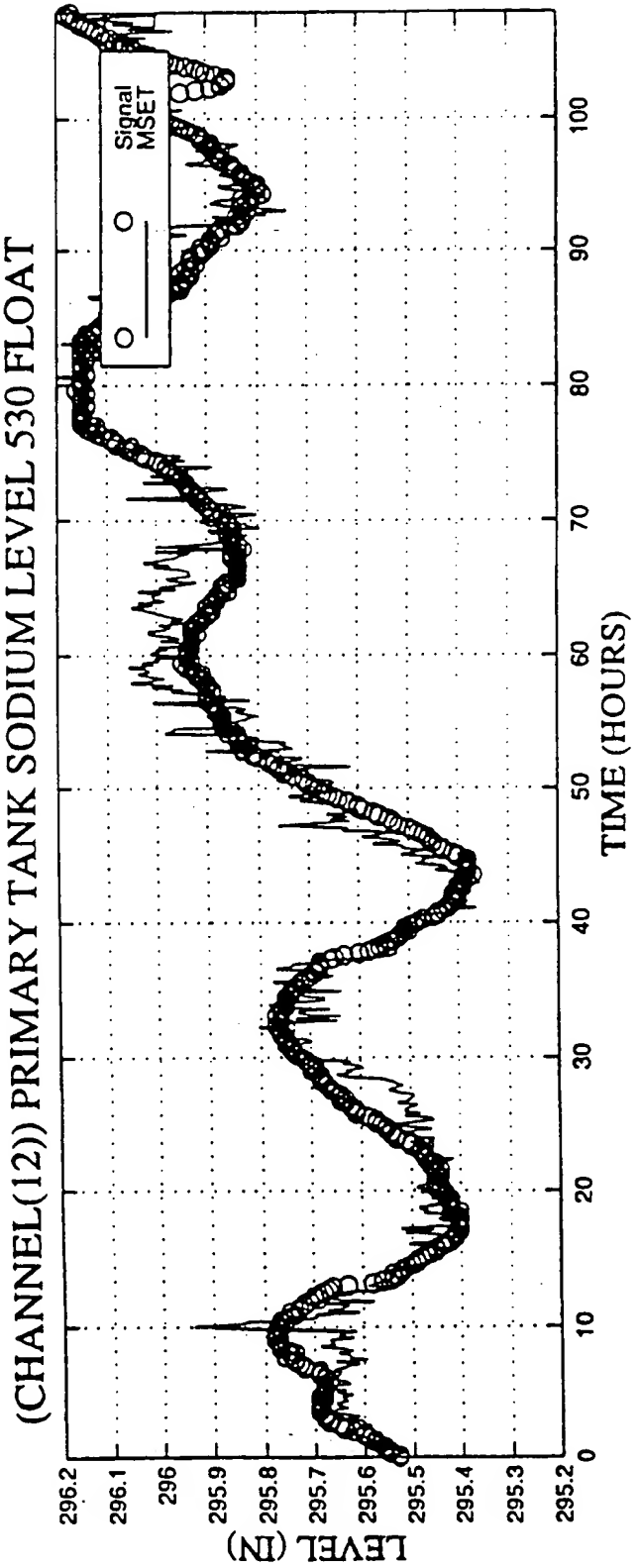

(CHANNEL(12)) PRIMARY TANK SODIUM LEVEL 530 FLOAT
Signal
MSET
LEVEL (IN)
296.2
296.1
296
295.9
295.8
295.7
295.6
295.5
295.4
295.3
295.2
0
10
20
30
40
50
60
70
80
90
100
TIME (HOURS)

FIG. 18A

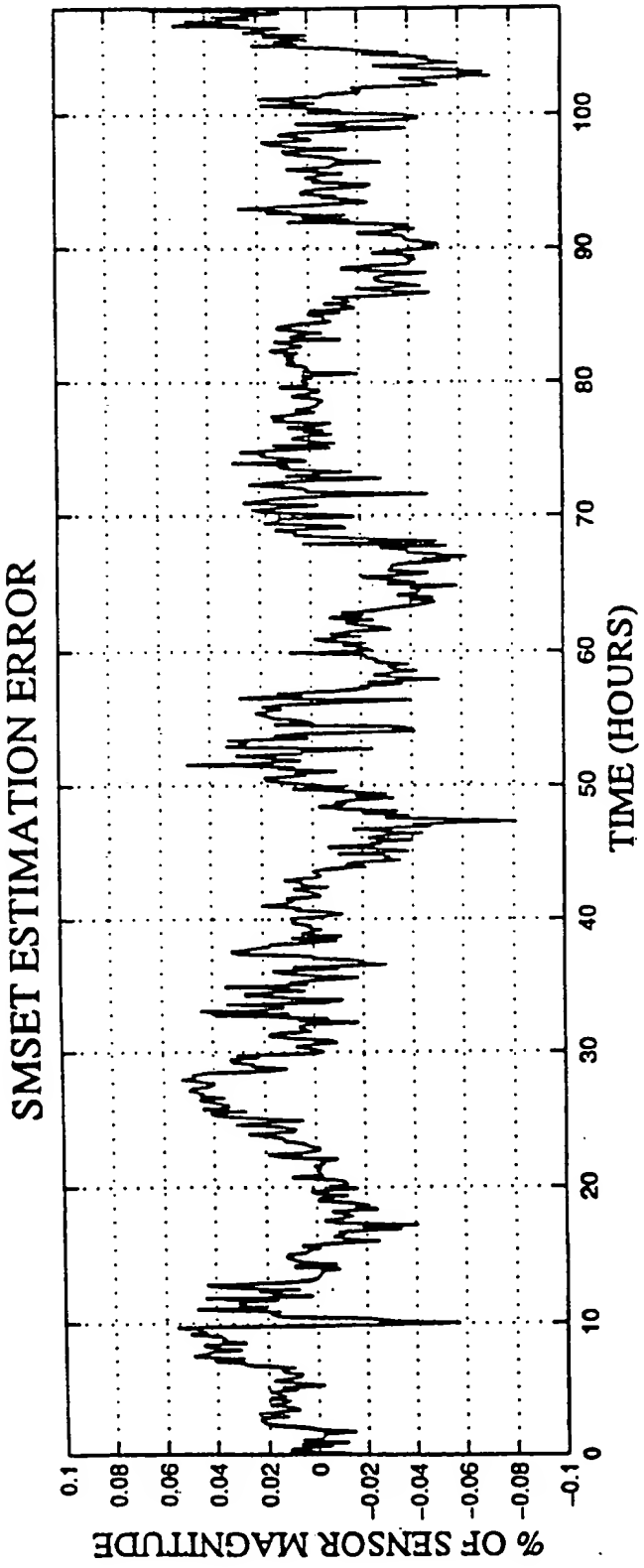

SMSET ESTIMATION ERROR
% OF SENSOR MAGNITUDE
0.1
0.08
0.06
0.04
0.02
0
-0.02
-0.04
-0.06
-0.08
-0.1
TIME (HOURS)
0
10
20
30
40
50
60
70
80
90
100

FIG. 18B

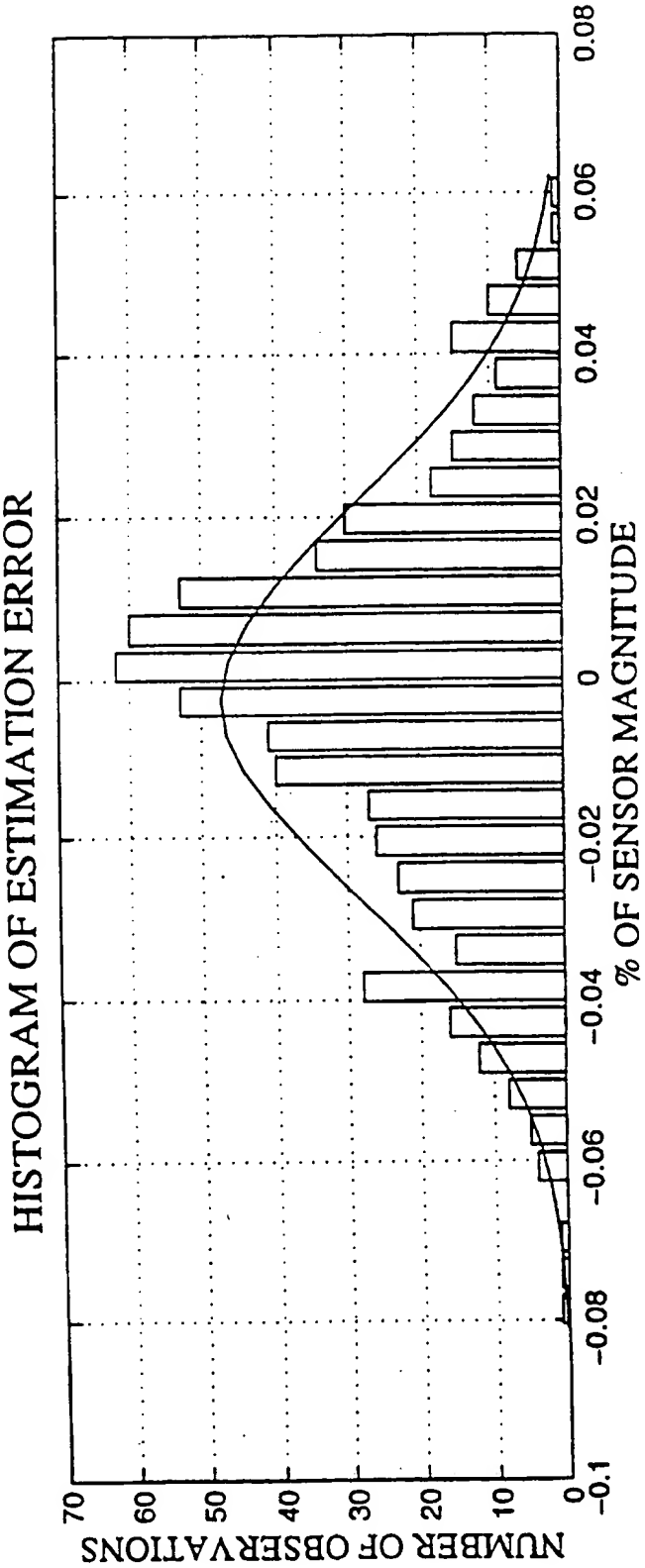

HISTOGRAM OF ESTIMATION ERROR
NUMBER OF OBSERVATIONS
% OF SENSOR MAGNITUDE
70
60
50
40
30
20
10
0
-0.1
-0.08
-0.06
-0.04
-0.02
0
0.02
0.04
0.06
0.08

FIG. 18C

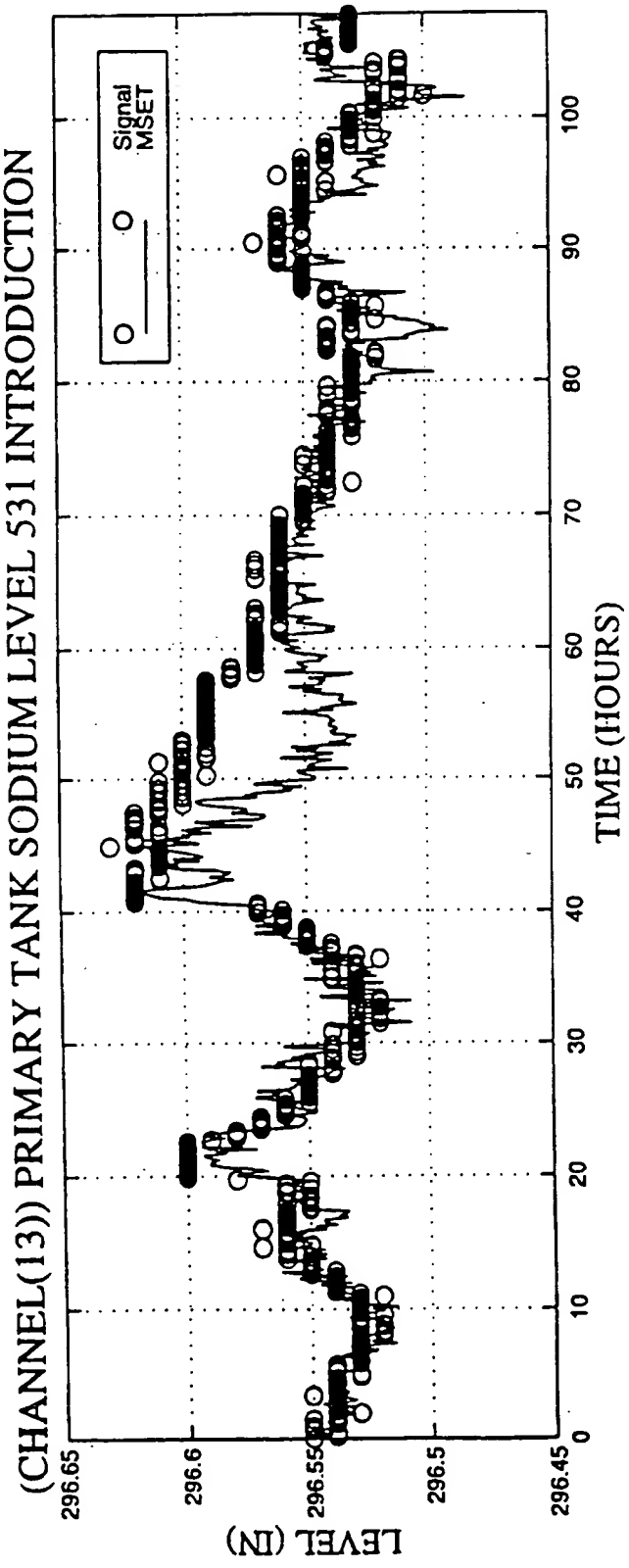

(CHANNEL(13)) PRIMARY TANK SODIUM LEVEL 531 INTRODUCTION
Signal
MSET
LEVEL (IN)
296.65
296.6
296.55
296.5
296.45
0
10
20
30
40
50
60
70
80
90
100
TIME (HOURS)

FIG. 19A

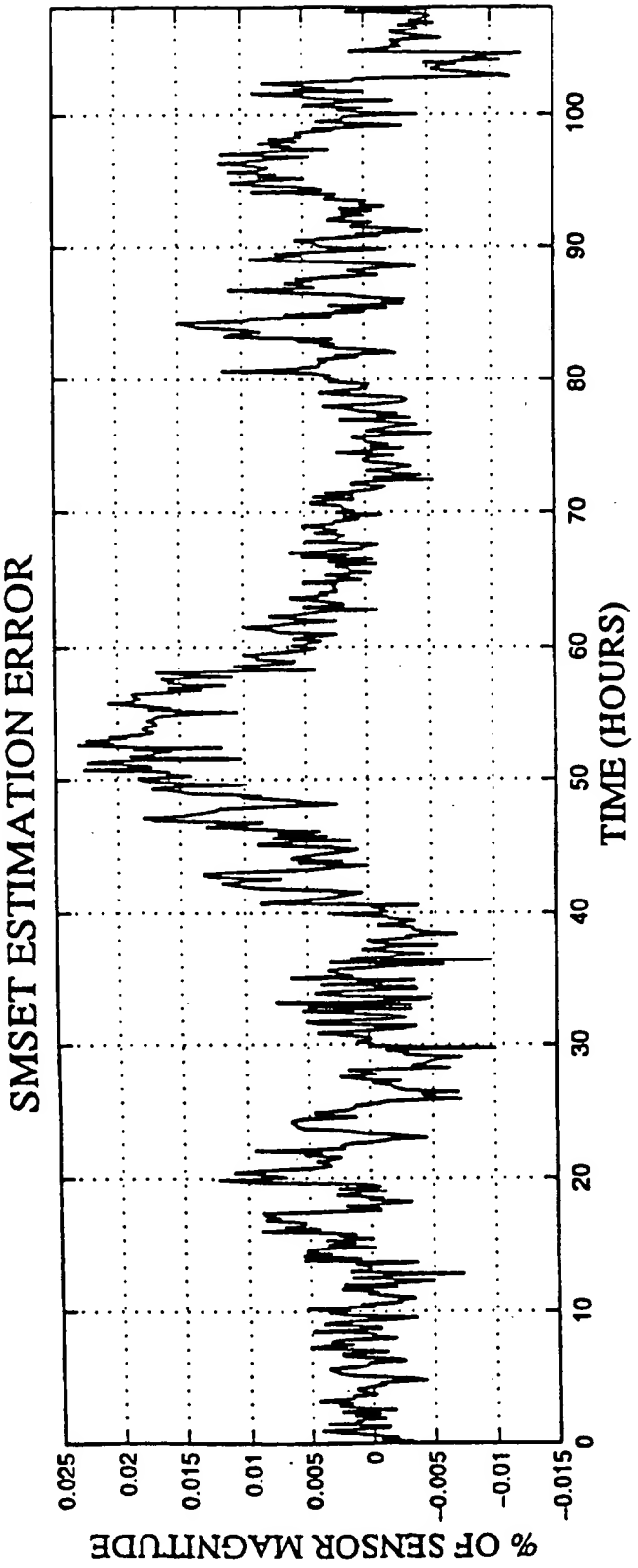

SMSET ESTIMATION ERROR
% OF SENSOR MAGNITUDE
0.025
0.02
0.015
0.01
0.005
0
-0.005
-0.01
-0.015
TIME (HOURS)
0
10
20
30
40
50
60
70
80
90
100

FIG. 19B

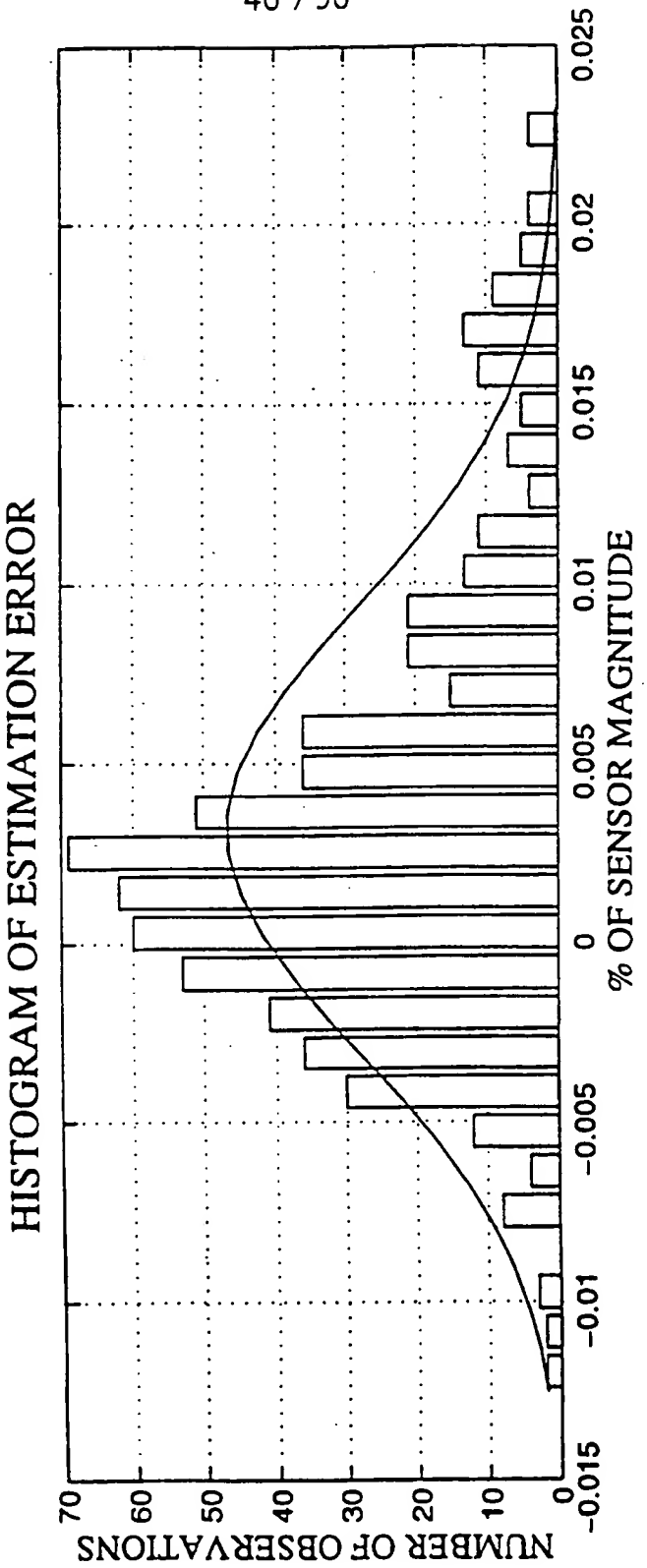

HISTOGRAM OF ESTIMATION ERROR
NUMBER OF OBSERVATIONS
% OF SENSOR MAGNITUDE
70
60
50
40
30
20
10
0
0.025
0.02
0.015
0.01
0.005
0
-0.005
-0.01
-0.015

FIG. 19C

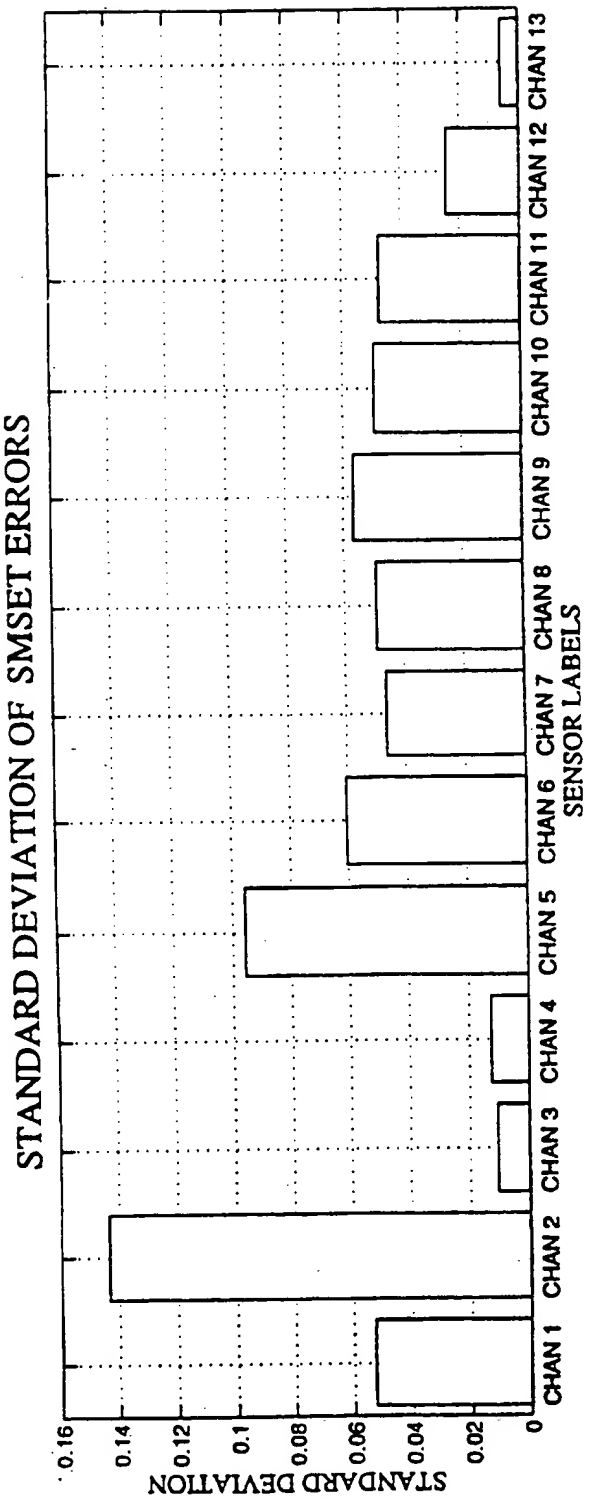

STANDARD DEVIATION OF SMSET ERRORS
STANDARD DEVIATION
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
CHAN 1
CHAN 2
CHAN 3
CHAN 4
CHAN 5
CHAN 6
CHAN 7
CHAN 8
CHAN 9
CHAN 10
CHAN 11
CHAN 12
CHAN 13
SENSOR LABELS

FIG. 20A

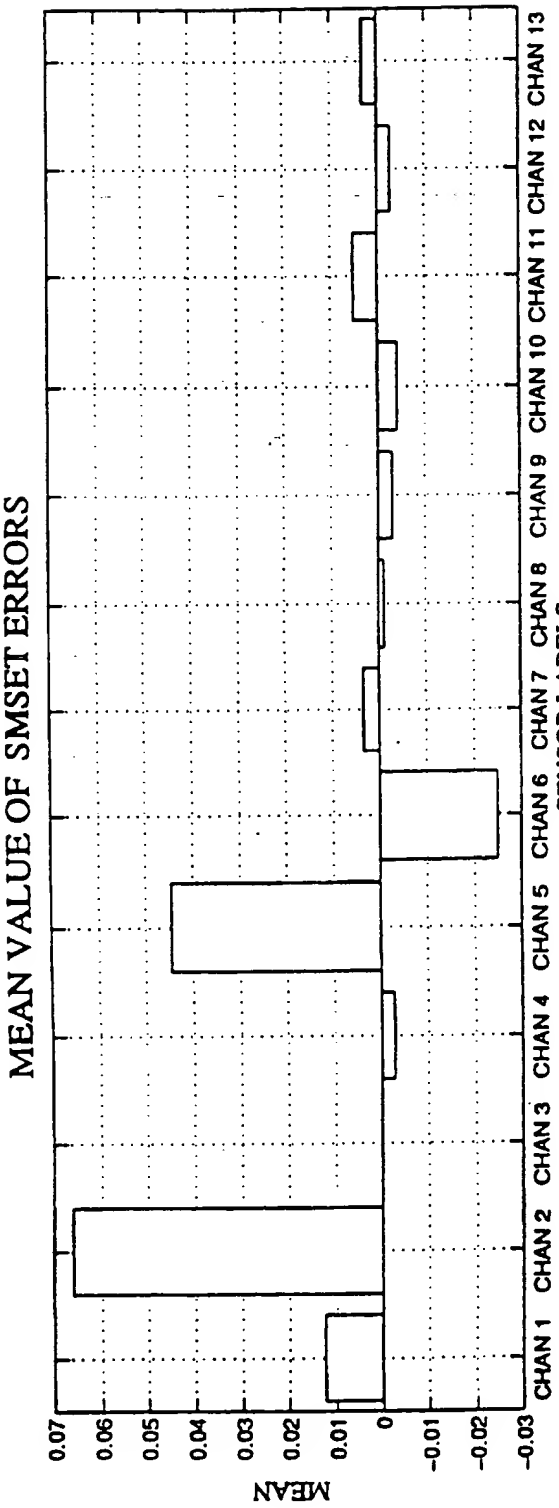

MEAN VALUE OF SMSET ERRORS
MEAN
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
-0.01
-0.02
-0.03
CHAN 1
CHAN 2
CHAN 3
CHAN 4
CHAN 5
CHAN 6
CHAN 7
CHAN 8
CHAN 9
CHAN 10
CHAN 11
CHAN 12
CHAN 13
SENSOR LABELS

FIG. 20B

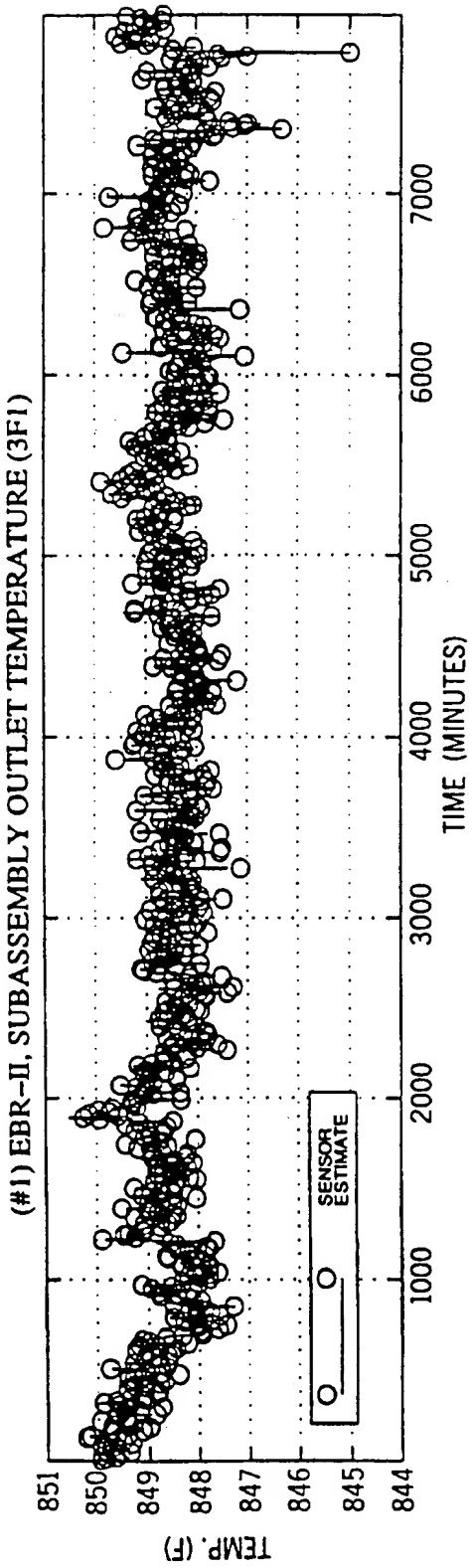

(#1) EBR-II, SUBASSEMBLY OUTLET TEMPERATURE (3F1)
TEMP. (F)
851
850
849
848
847
846
845
844
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)
SENSOR
ESTIMATE

FIG. 21A

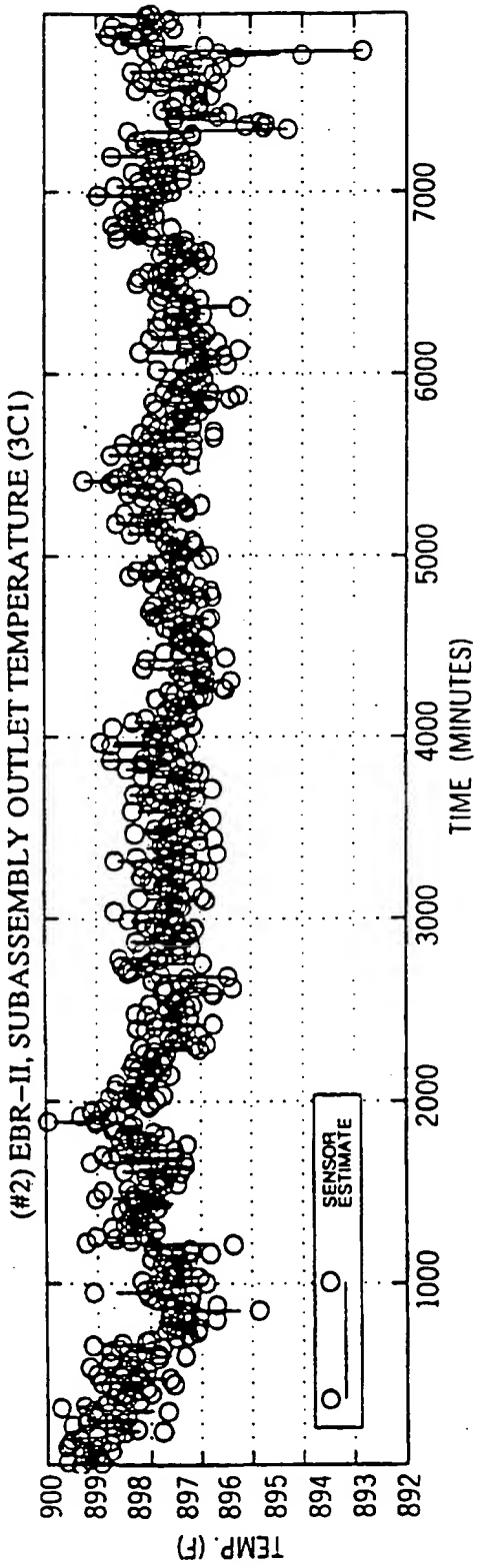

(#2) EBR-II, SUBASSEMBLY OUTLET TEMPERATURE (3C1)
TEMP. (F)
900
899
898
897
896
895
894
893
892
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)
SENSOR
ESTIMATE

FIG. 21B

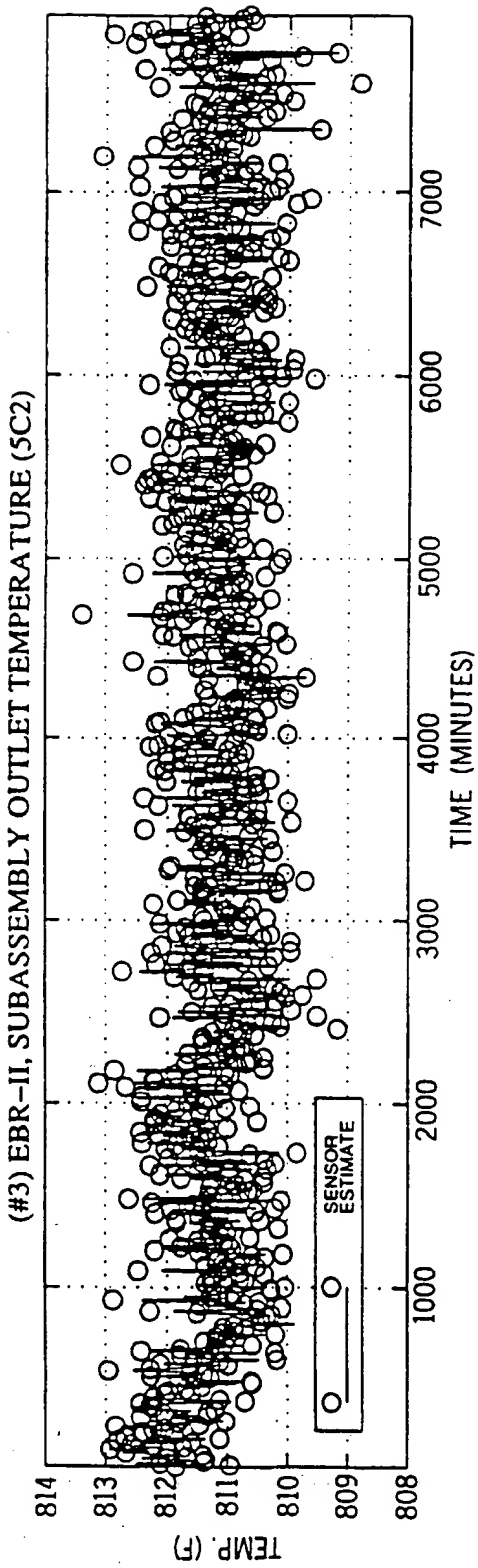

(#3) EBR–II, SUBASSEMBLY OUTLET TEMPERATURE (5C2)
TEMP. (F)
814
813
812
811
810
809
808
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)
SENSOR
ESTIMATE

FIG. 21C

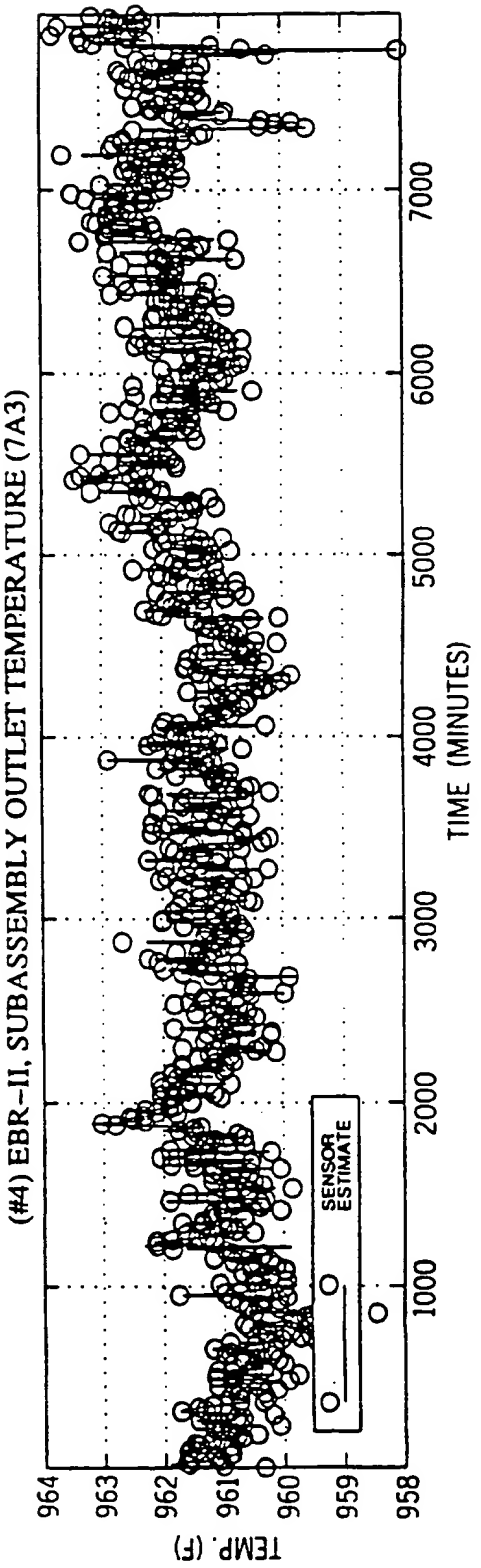

(#4) EBR-II, SUBASSEMBLY OUTLET TEMPERATURE (7A3)
TEMP. (F)
964
963
962
961
960
959
958
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)
SENSOR
ESTIMATE

FIG. 21D

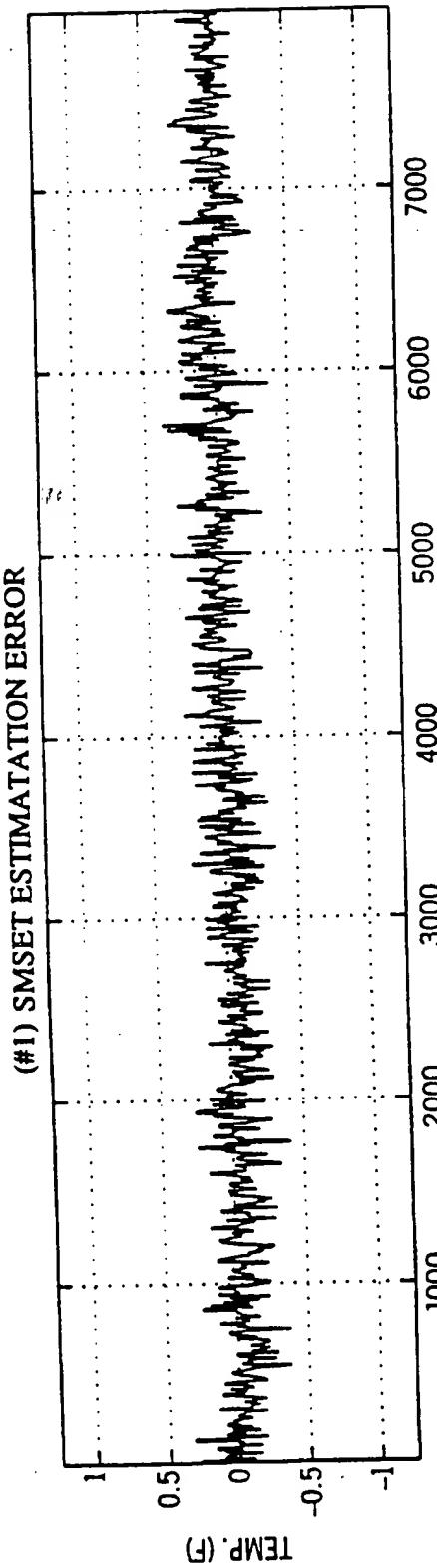

(#1) SMSET ESTIMATATION ERROR
TEMP. (F)
1
0.5
0
-0.5
-1
1000
2000
3000
4000
5000
6000
7000
TIME (MINUTES)

FIG. 22A

11. The method as defined in Claim 10 wherein the step of identifying a current state of the industrial process closest to the learned state includes forming a combination of the learned states to identify a true state of the industrial process.

12. The method as defined in Claim 10 further including the step of substituting an expected value for incomplete observations.

13. The method as defined in Claim 10 wherein the industrial data source is selected from the group consisting of an industrial manufacturing process, a utility operation, a business operation, an investment process, weather forecasting and a transportation system.

14. The method as defined in Claim 10 wherein the plurality of industrial data sources comprises a plurality of sensor pairs.

15. The method as defined in Claim 10 wherein the step of processing the modeled data comprises applying a SPRT process.

16. The method as defined in Claim 10 further including a step of determining time phase shift between the plurality of the time varying data being output.

5. The method as defined in Claim 4 wherein the step of determining phase shift comprises differentiating the cross correlation vector with respect to lag time between each pair of signals and performing an interpolation to compute the root of the differential of the cross correlation vector.

6. The method as defined in Claim 1 wherein the step of identifying a current state of the industrial process closest to the learned state includes forming a combination of the learned states to identity a true state of the industrial process.

7. The method as defined in Claim 6 further including the step of substituting an expected value for incomplete observations of the industrial process.

8. The method as defined in Claim 1 wherein the step of detecting a deviation from normal operation comprises applying a sequential probability ratio test to the modeled data.

9. The method as defined in Claim 1 wherein the step of detecting a deviation from normal operation comprises performing a pattern recognition analysis using computer means.

10. A method for monitoring at least one of an industrial process and an industrial data source, comprising the steps of:

generating time varying data from a plurality of industrial data sources;

determining learned states of a normal operational condition of the industrial process to use the learned states to generate expected values of the operating industrial process;

comparing the expected values to current values of the industrial process to identifying a current state of the industrial process closest to one of the learned states and generating a set of modeled data;

processing the modeled data to identify a pattern for the data and upon detecting a deviation from a pattern characteristic of normal operation, an alarm is generated.

What is Claimed is:

1. A method for monitoring an industrial process and industrial sensor, comprising the steps of:

generating time varying data from a plurality of industrial sensors;

processing the time varying data to effectuate optimum time correlation of the data accumulated from the plurality of industrial sensors;

searching the time correlated data to identify maximum and minimum values for the data, thereby determining a full range of values for the data from the industrial process;

determining learned states of a normal operational condition of the industrial process and using the learned states to generate expected values of the operating industrial process;

comparing the expected values to current actual values of the industrial process to identify a current state of the industrial process closest to one of the learned states ad generating a set of modeled data;

processing the modeled data to identify a pattern for the data ad upon detecting a deviation from a pattern characteristic of normal operation, a alarm is generated.

2. The method as defined in Claim 1 wherein the industrial process comprises a physical process.

3. The method as defined in Claim 1 wherein the industrial process comprises a financial process.

4. The method as defined in Claim 1 wherein the step of determining optimum time correlation comprises comparing pairs of sensor signals, each characteristic of a separate sensor and calculating a cross correlation vector over time, applying a low pass filter to remove noise from the cross correlation vector and determining phase shift between the sensor signals.

```
                    bu..i+1]='\0';
                    Dt[k][j]=(float) strtod(buf,NULL);
                    D[j][k]=(float) strtod(buf,NULL);
                    i=0;
                    flag=0;
                    j++;
                }
        }
        ColsD=k;
        fclose(Train);
```

```
    int num, flag, i, j;
    char ch, buf[16];
    num=0; flag=0; i=0; j=0;


    ch=fgetc(in);
    while((ch!='\n')&&(ch!=EOF)){
            num=(!(isspace(ch)));/*&&(ch!=',')&&(ch!='"'));*/
        /*  if (ch=='\n') {k++; RowsD=j; j=0;} */
            if (num==1){
                    buf[i]=ch;
                    i++;
                    flag=1;
            }
            if ((num==0)&&(flag==1)){

                    buf[i+1]='\0';
                    Sample[j]=(float) strtod(buf,NULL);
                    i=0;
                    flag=0;
                    j++;
            }
            ch=fgetc(in);
    }

    if (ch==EOF){
           /*       fclose(In);    */
            return (0);
    }
    else
            return (j);



at Ratio(float num1, float num2){

     if ((num1 == 0) || ( num2 == 0 ))
             return 0.0;
     else if (num1 >= num2)
             return num2/num1;
     else
             return num1/num2;

: AsIsD()

     int num, flag, i, j, k;
     char ch, buf[16];
     num=0; flag=0; i=0; j=0; k=0;
     while((ch=fgetc(Train))!=EOF){
             num=(!(isspace(ch))&&(ch!=',')&&(ch!='"'));
             if (ch=='\n') {k++; RowsD=j; j=0;}
             if (num==1){
                     buf[i]=ch;
                     i++;
                     flag=1;
             }
             if ((num==0)&&(flag==1)){
```

```
        if((Spos1>=BB)||(Sneg1>=BB)) sub1=1;
        if((Spos1<=AA)&&(Sneg1<=AA)) sub1=0;

        if((Spos2>=BB)||(Sneg2>=BB)) sub2=1;
        if((Spos2<=AA)&&(Sneg2<=AA)) sub2=0;



        fprintf(sprtpos1,"%f\n",Spos1);
        fprintf(sprtpos2,"%f\n",Spos2);
        fprintf(sprtneg1,"%f\n",Sneg1);
        fprintf(sprtneg2,"%f\n",Sneg2);

        fscanf(flow1,"%f\n",&temp1);
        fscanf(fast1,"%f\n",&temp2);

        if (sub1)
                fprintf(comb1,"%f\n",temp2);
        else
                fprintf(comb1,"%f\n",temp1);

        fscanf(flow2,"%f\n",&temp1);
        fscanf(fast2,"%f\n",&temp2);

        if (sub2)
                fprintf(comb2,"%f\n",temp2);
        else
                fprintf(comb2,"%f\n",temp1);
      if(fmod((double)(cnt+i+1),(double)per)==0){
        gotoxy(47,11);
        cprintf("%-3.3f ",((float)(cnt+i+1))/((float)Length)*100/2);
      }

}
 gotoxy(47,11);
 cprintf("%-3.3f ",100.0);
gotoxy(34,15);
textColor(WHITE,RED);
cprintf(" HIT ANY KEY ");
gotoxy(45,15);
getch();
fclose(sprtpos1); fclose(sprtneg1); fclose(fast1); fclose(error1);
                  fclose(comb1); fclose(flow1);
fclose(sprtpos2); fclose(sprtneg2); fclose(fast2); fclose(error2);
                  fclose(comb2); fclose(flow2);
fclose(in);
```

```
*************************************************************************/
*************************************************************************/
```

```
readinputSample(FILE *in)
```

```
        fgets(buf,13,error2);
        temp=atof(buf);
        mean2+=temp/calcLength;
}
fseek(error1,0L,0);
fseek(error2,0L,0);
variance1=0.0; variance2=0.0;
for (i=0;i<calcLength;i++){
        fgets(buf,13,error1);
        temp=atof(buf);
        variance1+=(temp-mean1)*(temp-mean1)/(calcLength-1);
        fgets(buf,13,error2);
        temp=atof(buf);
        variance2+=(temp-mean2)*(temp-mean2)/(calcLength-1);
}
fseek(error1,0L,0);
fseek(error2,0L,0);

/* Actual SPRT calculation and signal swapping logic */
sub1=0; sub2=0;
M1=FACTOR*sqrt(variance1);
g1=M1/variance1;
M2=FACTOR*sqrt(variance2);
g2=M2/variance2;
AA=log(BETA/(1-ALPHA));
BB=log((1-BETA)/ALPHA);
for(i=0;i<4;i++){
        if(i==0) fprintf(params,"%f\n",AA);
        if(i==1) fprintf(params,"%f\n",BB);
        if(i==2) fprintf(params,"%f\n",AA);
        if(i==3) fprintf(params,"%f\n",BB);
}
fclose(params);

Spos1=0; Spos2=0; Sneg1=0; Sneg2=0;

for (i=0;i<Length;i++){
  /*    fgets(buf,13,error2);
        temp=atof(buf);
  */

        fscanf(error1,"%f\n",&temp1);
        fscanf(error2,"%f\n",&temp2);

        if ((Spos1 <= AA)||(Spos1 >= BB)) Spos1=0;
        if ((Spos2 <= AA)||(Spos2 >= BB)) Spos2=0;
        if ((Sneg1 <= AA)||(Sneg1 >= BB)) Sneg1=0;
        if ((Sneg2 <= AA)||(Sneg2 >= BB)) Sneg2=0;

        steppos1= -g1*(M1/2 - temp1); stepneg1= -g1*(M1/2 + temp1);
        steppos2= -g2*(M2/2 - temp2); stepneg2= -g2*(M2/2 + temp2);

        Spos1 += steppos1; Sneg1 += stepneg1;
        Spos2 += steppos2; Sneg2 += stepneg2;

        if(Spos1 <= AA) Spos1=AA;  if(Sneg1 <= AA) Sneg1=AA;
        if(Spos2 <= AA) Spos2=AA;  if(Sneg2 <= AA) Sneg2=AA;

        if(Spos1 >= BB) Spos1=BB;  if(Sneg1 >= BB) Sneg1=BB;
        if(Spos2 >= BB) Spos2=BB;  if(Sneg2 >= BB) Sneg2=BB;
```

```
SPRT MODULE                                                    */
/*-------------------------------------------------------------*/


    /* Start the SPRT analysis..... */
    /* Calculate the variance of the estimate errors*/

    if (Length<calcLength) calcLength=Length;
    mean1=0.0; mean2=0.0;
    for (i=0;i<calcLength;i++){
            fgets(buf,13,errorl);
            temp=atof(buf);
            mean1+=temp/calcLength;
```

Appendix D

Computer software for performing pattern recognition by detecting onset of degradation in noisy signals.

and

```
% function [z, norm_dist] = vprprod (x, y, alpha, beta);
%
%  This function performs the nonlinear operation on matrices x and y at the
%  heart of the MSET.  It uses the following nonlinear
%  operator to compare vectors in the matrices:
%      f(dist) = 1 / (1 + alpha dist^beta),
%  where dist is the normalized distance between a vector in y and each
%  observation vector in x.
%
%  If the constant alpha and beta are not specified, default values of 1 are
%  used for both of them.
%
%  This version of vprprod returns the norm of the distance between the
%  each vector in y and the exemplar vetors in x.  The vector distances are
%  normalized with respect to the sum of the distance vectors.
%
%
%

function [z, norm_dist] = vprprod (x, y, alpha, beta);

if (nargin == 2)
     alpha = 1;
     beta = 1;
end

[Nx,Mx] = size(x);
[Ny,My] = size(y);


if (Mx ~= Ny)
     fprintf('\n\n in vprprod(x,y), the number of columns in x must\n');
     fprintf(' equal the number of rows in y\n\n');
     str=['columns in x: ', num2str(Mx), ...
     ' , and the number of rows in y: ', num2str(Ny)];
     error(str);
end

% Initialize arrays.
z = zeros(Nx,My);
norm = zeros(Nx,My);
yi = zeros(Nx,Mx);
dist = zeros(Nx,1);
tmp = zeros(Mx,Nx);

for i = 1:My

     % Expand each column of y into a Nx by Ny (= Nx by Mx) matrix.
```

```
% Reset Y matrix to input matrix.
Y = Input(:,2:size(Input,2))';

% Calculate error.
Err = Y(prt_cols,:) - X_hat;

% Calculate rms error for variables specified by prt_cols.
Erms = diag(Err * Err');
Erms = sqrt(Erms' ./ (N - M - 1));

% Add timestamp to X_hat and Err arrays and transpose X_hat and Err arrays.
X_hat = [Time'; X_hat]';
Err = [Time'; Err]';

if (print_flag >= 1)
  btm=fix(clock);
  fprintf('\n\n Total time to complete MSET using %d training vectors\n', ...
           size(D,2));
  fprintf(' to analyze %d observations with %d variables each: ', N, M);
  fprintf('%6.2f mins\n\n', toc/60);
end
```

```
X_hat = zeros(length(prt_cols),N);
tmp = zeros(M,1);
Erms = zeros(length(prt_cols),1);
Err = zeros(length(prt_cols),N);
DtY = zeros(M-length(out_cols),1);
W = zeros(N,1);

% Main loop.  Calculate estimate vectors (X_hat) for each input state
% vector Y.
for i=1:N

    DtY = vprprod(D_proj', Y_proj(:,i), alpha, beta);
    W = DDi * DtY;
       WW2(:,i)=W;
    %W = W ./ sum(W);
    ii=find(W<thrsh);
    if(length(ii)>0)
       W(ii)=zeros(length(ii),1);
    %  W = W./sum(W);
    end
       WW(:,i)=W;
    tmp = D * W;
    X_hat(:,i) = tmp(prt_cols);

    % Print message if print_flag is set.
    if (print_flag >= 1)
        if ((rem(i,floor(N/100)) == 0) | (i==1))
            fprintf('\b\b\b');
            fprintf('%3.0f', i/N*100);
        end
    end

end

% Scale the X_hat matrix from the range 0:1 back to the range specified by
% the rangeL and rangeU vectors, if necessary.
if (length(rangeL) == M) & (length(rangeU) == M)
    % Scale each observation vector in the training matrix.
    %NNN=length(rangeU(prt_cols));
    %rU=rangeU(prt_cols); rL=rangeL(prt_cols);
    %X_hat=(rU(:)*ones(1,N)-...
    %           rL(:)*ones(1,N))...
    %         .*  X_hat + rL(:)*ones(1,N);
    for i = 1:length(prt_cols)
        X_hat(i,:) = (rangeU(prt_cols(i)) - rangeL(prt_cols(i))) * ...
                     X_hat(i,:)  +  rangeL(prt_cols(i));
    end
end

% Shift the X_hat matrix back to the range of the data by adding rangeL to
% the data, if necessary.
if (length(rangeL) == M) & (rangeU == 0)
    % Shift each observation vector in the training matrix.
    NNN=length(prt_cols);
    X_hat=X_hat + rangeL(prt_cols)*ones(1,NNN);
    %for i = 1:length(prt_cols)
    %    X_hat(i,:) = X_hat(i,:)  +  rangeL(prt_cols(i));
    %end
end
```

```
if nargin == 7, alpha=1; beta=1; p_flag=0; end
if nargin == 8, beta=1; p_flag=0; end
if nargin == 9, p_flag=0; end
if prt_cols == 0, prt_cols=1:M; end
if alpha == 0, alpha=1; end
if beta == 0, beta=1; end

% Set print flag for debug printout.
print_flag = p_flag;

% Print message if print_flag is set.
if (print_flag >= 1)
     fprintf('\n Starting MSET ........\n\n');
end

% Scale the input matrix to the range 0:1, if the rangeL and rangeU
% vectors are specified.
if (length(rangeL) == M) & (length(rangeU) == M)
     % Scale each observation vector in the training matrix.
          Y=(Y-rangeL'*ones(1,N))./...
                    (rangeU'*ones(1,N)-rangeL'*ones(1,N));
     %for i = 1:M
     %     Y(i,:) = (Y(i,:) - rangeL(i)) / (rangeU(i) - rangeL(i));
     %end

     % Print message if print_flag is set.
     if (print_flag >= 1)
          fprintf(' Data is scaled to 0:1 range....\n\n');
     end
end

% Shift the training matrix by the values specified in rangeL, if the rangeL
% vector is specified and rangeU is 0.
if (length(rangeL) == M) & (rangeU == 0)
     % Shift each observation vector in the training matrix.
      Y = Y - rangeL'*ones(1,N);
     %for i = 1:M
     %     Y(i,:) = Y(i,:) - rangeL(i);
     %end

     % Print message if print_flag is set.
     if (print_flag >= 1)
          fprintf(' Data is shifted by the values in rangeL....\n\n');
     end
end

% If the are output columns specified, then remove these variables from the
% D and Y matrices.
D_proj = D;
Y_proj = Y;
if (out_cols == 0)
     D_proj(out_cols,:) = [];
     Y_proj(out_cols,:) = [];
end

% Print message if print_flag is set.
if (print_flag >= 1)
     fprintf(' percent complete:   0');
end
```

```
% function [Ermss, X_hat, Err,WW2,WW] = mset (Input, rangeL, rangeU, D, DDi, ...
%          out_cols, prt_cols, alpha, beta, p_flag,thrsh);
%
%     This function estimates the state of a system using the (MSET).  It
%     allows scaling of the input data to the range
%     0 - 1 and the use of the algorithm in a predictive mode.
%
% Function outputs:
%      Ermss     = Root mean squared errors for variables specified by
%                  prt_cols.
%      X_hat     = Estimated states for variables specified by prt_cols.
%      Err       = Estimation error for variables specified by prt_cols.
%
% Function inputs:
%      Input     = input state matrix: an N by M+1 array with the first column
%                  containing a timestamp.  The array contains n observation
%                  vectors, with m dependent variables in each vector.
%      rangeL    = Length M vector, where each element sets the lower limit of
%                  data for a corresponding variable.
%      rangeU    = Length M vector, where each element sets the upper limit of
%                  data for a corresponding variable.
%      D         = Training set, an M by N array.
%      DDi       = Inverse of the similarity matrix (from auto_trainmsa).
%      out_cols  = Optional vector specifying which columns represent output
%                  variables predicted by the algorithm.  These variables are
%                  not included in the evaluation of W.
%      prt_cols  = Optional vector specifying which columns to keep in the
%                  output.
%      alpha     = Multiplicative factor in the vprprod nonlinear operator.
%      beta      = Power factor in the vprprod nonlinear operator.
%      p_flag    = Print/don't print run progress data for 0/1.
%      thrsh     = Cutoff threshold for weighting vector W (default 0)
% The rangeL and rangeU vectors are used to specify the anticipated range of
% the variables.  Data is linearly scaled from the ranges specified by these
% vectors to the range 0:1.  If the rangeL vector specifies the lower limit of
% the data while rangeU is 0, then the code will shift the data by subtracting
% the rangeL values from each element in a corresponding column of the data.
% If both of the rangeL and rangeU vectors are set to a scalar value of 0,
% the data will not be scaled.

function [Ermss, X_hat, Err,WW2,WW] = mset (Input, rangeL, rangeU, D, DDi, ...
         out_cols, prt_cols, alpha, beta, p_flag, thrsh);
if (nargin < 11), thrsh=0.0; end
tic

% Case in which not enough inputs are specified.
if nargin < 5
     error('Required arguments are: Input, rangeL, rangeU, D and DDi\n\n');
end

% Remove time stamp from input matrix and transpose input matrix.
Time = Input(:,1);
Y = Input(:,2:size(Input,2))';
[M, N] = size(Y);

% If out_cols, prt_cols, alpha, beta, and p_flag are not specified, pick
% default values for them.
if nargin == 5, out_cols=0; prt_cols=1:M; alpha=1; beta=1; p_flag=0; end
if nargin == 6, prt_cols=1:M; alpha=1; beta=1; p_flag=0; end
```

Appendix C

Computer software for modeling behavior for examples of the operating states of the commercial system.

```
      normvideo();
      textLook(YELLOW,BLUE);


}

void textLook(int txt, int bg){

      textbackground(bg);
      textcolor(txt);
}
void NewName(int xpos, int ypos, int whc)
{
      int i, c, maxi;
      char temp[4];

      textLook(LIGHTGRAY,BLACK);
      gotoxy(xpos-12,ypos);
      printf("              ");
      gotoxy(xpos-12,ypos);
      i=0;
      maxi=-1;
      while ((c=getche())!=13){
            if (c==8) {i=i-1;
                  if (i<0) i=0;
            }
            else{
                  maxi=max(i,maxi);
            switch (whc){
                  case 1: training[i]=c; break;
                  case 2: input[i]=c;    break;
                  case 3: temp[i]=c; break;
                  case 4: temp[i]=c;break;
                  case 5: temp[i]=c;break;
                  default: break;
            }
            i++;
            }

      }
      switch (whc){
                  case 1: training[maxi+1]='\0';break;
                  case 2: input[maxi+1]='\0';   break;
                  case 3: temp[maxi+1]='\0';
                        FACTOR=(float) atof(temp); break;
                  case 4: temp[maxi+1]='\0';
                        F1=atoi(temp); break;
                  case 5:
                        temp[maxi+1]='\0';
                        F2=atoi(temp); break;
                  default: break;
            }

             /*   F1=F1-1;
                  F2=F2-1;       */
}
```

```
        }
        if (ch==72){
              yy=yy-2;
              if (yy<11)
                    yy=21;
        }

        if ((ch==8)&&(yy!=21)){

              if (yy==11)
                    NewName(xx,yy,1);     /*  1=training */
              if (yy==13)
                    NewName(xx,yy,2);     /*  2=input    */
              if (yy==15)
                    NewName(xx,yy,3);     /*  3=output   */
              if (yy==17)
                    NewName(xx,yy,4);
              if (yy==19)
                    NewName(xx,yy,5);
              yy=yy+2;
        }
        if (yy==21)
              xx=42;
        else
              xx=53;

        if ((yy==21)&&(ch==13)) stop=1;
        /*      TESTING CONTENTS        */

        /*
        gotoxy(1,22);
        cprintf("                              ");
        gotoxy(1,22);
        cprintf("training = %s",training);
        gotoxy(1,23);
        cprintf("                              ");
        gotoxy(1,23);
        cprintf("input = %s\n",input);
        gotoxy(1,24);
        cprintf("                              ");
        gotoxy(1,24);
        cprintf("output = %s\n",output);
        */
        gotoxy(xx,yy);


    }

}

void InitialScreen()
{
      textbackground(CYAN);
      clrscr();

      textLook(RED,BLACK);
      highvideo();
      gotoxy(24,2);
      cprintf("                                      ");
      gotoxy(24,3);
      cprintf(" MSET SIGNAL VALIDATION SYSTEM ");
      gotoxy(24,4);
      cprintf("                                      ");
```

```
void GetSettings()
{
      int ch, yy, xx, stop;
      char *buff;

      /* default F1 and F2 : columns for flow 1 and flow 2*/
      F1=1;
      F2=2;

      textLook(YELLOW,BLUE);
      gotoxy(8,7);
     cprintf(" The default input and output file names are listed below.
");
      gotoxy(8,8);
     cprintf(" Use %c and %c to move to selection, <backspace> to change name.
",
                  24,25);
        gotoxy(22,11);
        textLook(BLACK,LIGHTGRAY);
        cprintf(" 1) training data:");
        gotoxy(40,11);
        textLook(LIGHTGRAY,BLACK);

        cprintf(" %s     ",training); /* use gotoxy(51,11) */
        gotoxy(22,13);
        textLook(BLACK,LIGHTGRAY);
        cprintf(" 2) input data    :");
        gotoxy(40,13);
        textLook(LIGHTGRAY,BLACK);
        cprintf(" %s        ",input); /* use gotoxy(51,13) */
        gotoxy(22,15);
        textLook(BLACK,LIGHTGRAY);
        cprintf(" 3) SFM           :");
        gotoxy(40,15);
        textLook(LIGHTGRAY,BLACK);
        cprintf(" 2                "); /* use gotoxy(51,15) */
        gotoxy(22,17);
        textLook(BLACK,LIGHTGRAY);
        cprintf(" 4) Column flow 1:");
        gotoxy(40,17);
        textLook(LIGHTGRAY,BLACK);
        cprintf(" 1                ");
        textLook(BLACK,LIGHTGRAY);
        gotoxy(22,19);
        cprintf(" 5) Column flow 2:");
        gotoxy(40,19);
        textLook(LIGHTGRAY,BLACK);
        cprintf(" 2                ");
        gotoxy(37,21);
        textLook(RED,LIGHTGRAY);
        cprintf(" done "); /*use goto(42,21) */

        /* make selection */
        gotoxy(53,11);
        yy=11;
        xx=53;
        stop=0;
        while (stop!=1){
            ch=getch();

            if (ch==80){
                  yy=yy+2;
                  if (yy>21)
                        yy=11;
```

```
      i=0;
      while((c=getc(f1))!='\n'){


}
*/
int ChooseTraining()
{
       int ch, yy, xx, stop;
       char *buff;
       textLook(YELLOW,BLUE);
       gotoxy(8,7);
      cprintf(" There are four different ways of extracting training data from
");
       gotoxy(8,8);
      cprintf(" the training file, ");
      gotoxy(28,8);
      textLook(13,BLUE);
      cprintf("%-12s",training);
      gotoxy(40,8);
      textLook(YELLOW,BLUE);
      cprintf(", choose one below:                ");
         gotoxy(30,11);
         textLook(BLACK,LIGHTGRAY);
         cprintf(" 1) Use File as is ");
         gotoxy(30,13);
         cprintf(" 2) MinMax method  ");
         gotoxy(30,15);
         cprintf(" 3) Equally Spaced ");
         gotoxy(30,17);
         cprintf(" 4) Specify States ");

         /* make selection */
         gotoxy(48,11);
         yy=11;
         xx=48;
         stop=0;
         while (stop!=1){
             ch=getch();

             if (ch==80){
                   yy=yy+2;
                   if (yy>17)
                         yy=11;
             }
             if (ch==72){
                   yy=yy-2;
                   if (yy<11)
                         yy=17;
             }
             if (ch==13) stop=1;
             /*     TESTING CONTENTS        */

             /*
             gotoxy(1,22);
             cprintf("                              ");
             gotoxy(1,22);
             cprintf("choice = %d",((yy-9)/2));
             */
             gotoxy(xx,yy);


         }
         return ((yy-9)/2);
}
```

```
        cprintf("          Extending training matrix.....     ");
        extendD(ex);
      }
}
void UserSpecified(){
      InitialScreen();
      gotoxy(14,8);
      cprintf(" This feature is not available in the DEMO version ");
      gotoxy(14,9);
      cprintf(" using MinMax  (hit a key to continue).....        ");
      getch();
      MinMax();
}

void extendD(float ex){
      int i,j;
      float mn[MAXSENS/*RowsD*/], std[MAXSENS/*RowsD*/],tmp;
      /**** Initializing**************/
      for(i=0;i<RowsD;i++){
            mn[i]=0.0;
            std[i]=0.0;
      }

      /*****CAlculating the mean ******/
      for(i=0;i<ColsD;i++)
            for(j=0;j<RowsD;j++)
                  mn[j]+=D[j][i];
      for(j=0;j<RowsD;j++)
            mn[j]/=ColsD;

      /**** Calculating the standard deviation*****/
      for(i=0;i<ColsD;i++)
            for(j=0;j<RowsD;j++)
                  std[j]+=(D[j][i]-mn[j])*(D[j][i]-mn[j]);
      for(j=0;j<RowsD;j++)
            std[j]/=(ColsD-1);

      /**** Extending the range of the training matrix***/
      for(i=0;i<ColsD;i++){
            for(j=0;j<RowsD;j++){
                  if(D[j][i]<=mn[j]){
                        D[j][i]-=std[j]*ex;
                        Dt[i][j]-=std[j]*ex;
                  }
                  else{
                        D[j][i]+=std[j]*ex;
                        Dt[i][j]+=std[j]*ex;
                  }
            }
      }

}
int lengthFile(FILE *fl){
      char c;
      int i;
      i=0;
      while((c=getc(fl))!=EOF)
            if(c=='\n') i++;
      fseek(fl,0L,0);
      return (i);
}
/*
int widthFile(FILE *fl){
      char c;
      int i;
```

```
void EquallySpaced(){

      int L,W,i,j,k,count,num,c,stop;
      float ex;

      InitialScreen();
      textLook(YELLOW,BLUE);
      fseek(Train,0L,0);

      gotoxy(17,8);
      cprintf("     train using the full file? (y/n):       ");
      gotoxy(56,8);
      c=getche();

      if(c=='n'){
            gotoxy(17,8);
            cprintf("      How many lines into the file:          ");
            gotoxy(53,8);
            cscanf("%d",&L);
      }
      else{
            L=lengthFile(Train);
            fseek(Train,0L,0);
      }
      stop=0;
      while (!stop){
       gotoxy(14,8);/*                                          */

       cprintf("    Enter numer of training vectors (max 132):     ");
       gotoxy(61,8);
       cscanf("%d",&num);
       if(num>132) stop=1;
      }
      gotoxy(14,8);
      cprintf("        Getting vectors from training file....     ");
      count=floor(((float)L)/((float) num));
      if(count<1){ count=1; L=num;}
      ColsD=num;
      j=0;
      for (i=0;i<L;i++){
            if((i%count)==0){
                  W=readinputSample(Train);
                  RowsD=W;
                  for(k=0;k<RowsD;k++){
                        D[k][j]=Sample[k];
                        Dt[j][k]=Sample[k];
                  }
                  j++;
            }
      }
      gotoxy(17,8);/*                                         */
      cprintf("        Extend training range (y/n):       ");
      gotoxy(56,8);
      /*cscanf("%c",&c);*/
      getche();
      c=getche();
      if(c=='y'){
       gotoxy(17,14);/*                                       */
       cprintf(" How many standard deviations from the mean: ");
       gotoxy(17,15);
       cprintf(" (typical value = 3) Enter value =           ");
       gotoxy(53,15);
       cscanf("%f",&ex);
       gotoxy(17,17);
```

```
numPairs=2*W;
didSwitch=1;
while(didSwitch){
      numPairs--;
      didSwitch=0;
      for(i=0;i<numPairs;i++){
            if(time_pts[i]>time_pts[i+1]){
                  tmp=time_pts[i];
                  time_pts[i]=time_pts[i+1];
                  time_pts[i+1]=tmp;
                  didSwitch=1;
            }
      }
}
/******* getting unique time points ******/                    */
gotoxy(17,8);/*                                                  */
cprintf("    Removing redundant time points.....               ");

j=0;
for(i=1;i<2*W;i++){
      if(time_pts[i-1]!=time_pts[i]){
            j++;
            time_pts[j]=time_pts[i];
      }
}
ColsD=j;
RowsD=W;
j=0;                                                             */
gotoxy(17,8);/*                                                  */
cprintf("    Creating training matrix.....                     ");

for(i=0;i<L;i++){
      W=readinputSample(Train);
      if(i==time_pts[j]){
            for(k=0;k<RowsD;k++){
                  D[k][j]=Sample[k];
                  Dt[j][k]=Sample[k];
            }
            j++;
      }
}                                                                */
gotoxy(17,8);/*                                                  */
cprintf("        Extend training range (y/n):                    ");
gotoxy(56,8);
/*cscanf("%c",&c);*/
getche();
c=getche();
if(c=='y'){                                                      */
 gotoxy(17,14);/*                                                */
 cprintf(" How many standard deviations from the mean: ");
 gotoxy(17,15);
 cprintf(" (typical value = 3) Enter value =                     ");
 gotoxy(53,15);
 cscanf("%f",&ex);
 gotoxy(17,17);
 cprintf("          Extending training matrix.....               ");
 extendD(ex);
}

}

/*****************************************************************/
/*****************************************************************/
```

```
/*****************************************************************/
/*    MINIMAX TRAINING MODULE                                    */
/*****************************************************************/



void MinMax(){

      int L,W,j,i,k,numPairs,didSwitch;
      int time_pts[MAXSENS*2],c;
      /*int min_time_pts[MAXSENS];*/
      float minmax[2][MAXSENS],tmp,ex;

      InitialScreen();
      textLook(YELLOW,BLUE);
      fseek(Train,0L,0);

      gotoxy(17,8);
      cprintf("     train using the full file? (y/n):       ");
      gotoxy(56,8);
      c=getche();
      /*cscanf("%c",&c);*/

      if(c=='n'){
            gotoxy(17,8);
            cprintf("      How many lines into the file:          ");
            gotoxy(53,8);
            cscanf("%d",&L);
      }
      else{
            L=lengthFile(Train);
            fseek(Train,0L,0);
      }


      W=readinputSample(Train);

      gotoxy(17,8);/*                                             */
      cprintf("    Finding max and min for all sensors...   ");
      for(j=0;j<W;j++){
            minmax[0][j]=Sample[j];
            minmax[1][j]=Sample[j];
            time_pts[j]=0;
            time_pts[W+j]=0;
      }
      /***** Finding time points for each sensor ******/
      for(i=1;i<L;i++){
            W=readinputSample(Train);
            for(j=0;j<W;j++){
                  if(Sample[j]>=minmax[0][j])
                        time_pts[j]=i;
                  if(Sample[j]<=minmax[1][j])
                        time_pts[W-1+j]=i;
            }
      }
      fseek(Train,0L,0);
      gotoxy(17,8);/*                                           */
      cprintf("    Sorting time points.....          ");

      /**********************/
      /********sorting  *******/
```

Appendix B

Computer software for producing an optimal training set derived by searching signal information during a training period to construct training vectors including highest and lowest points for signals under surveillance.

```
void fliplr(float a[], int size)
{
 int i,j;
 float temp;

 j=floor(size/2);
 for(i=0;i<j;i++)
  { temp=a[i];
    a[i]=a[size-i-1];
    a[size-i-1]=temp;
  }
}

void der(float y[], float diffy[], int size)
{
  int i;
  float diff;

  for(i=0;i<size-1;i++)
     diffy[i] = y[i+1]-y[i];

}

void roots(float diffxc[], int size, float *root)
{
 int i,j;
 float term;

 if(diffxc[0]*diffxc[size-1] > 0)
   { printf("The cross_correlation is not unimodal.\n");
     exit(0);
   }
 else
   { (*root)=0.0;
     for(i=0;i<size-1;i++)
       {  term=i;
          for(j=0;j<size-1;j++)
           if((i-j)>0)
             term=term*((-diffxc[j])/(diffxc[i]-diffxc[j]));
          (*root) = (*root) + term;

       }
   }
}


void prn_info(char *pgm_name)
{
 printf("\n%s%s%s\n",
     "Usage: ", pgm_name, " infile1 infile2 infile3 outfile");
 printf("\n");
 printf("Note that infile1 & infile2 are reference and input signal");
 printf(" respectively.  infile3 should contain the coefficients of");
 printf(" the LPF, and outfile receives the phase-equalized input signal.\n");
}
```

```
{ ph=fil_xc[0];
  j=0;
  for(i=0;i<fil_xcsize;i++)
    if(fil_xc[i]>ph)
     { ph=fil_xc[i];
       j=i;
     }

  ph=j-(fil_xcsize+1)/2;
  printf("The shift is %d\n",ph);
}

/* Find the shift, using derivative technique. */
if(method==1)
 { der(fil_xc,difffil_xc,fil_xcsize);
   roots(difffil_xc,fil_xcsize-1,&root);
   ph=root-(fil_xcsize+1)/2;
   printf("The shift is %f\n",ph);
 }

/*  Phase-equalize the input (f2) relative to reference (f1) */
/* If ph>0, f2 leads f1; if ph<0, f2 lags f1. */
if(ph>0)
 { for(i=ph;i<N;i++)
     f2[i-ph] = f2[i];
   for(i=N-ph;i<N;i++)
     f2[i]=0;
 }

if(ph<0)
 { ph = -ph;
   for(i=N-1;i>=ph;i--)
    f2[i] = f2[i-ph];
   for(i=0;i<ph;i++)
     f2[i]=0;
 }

for(i=0;i<N;i++)
  fprintf(outfile,"%f\n",f2[i]);


fclose(infile1);
fclose(infile2);
fclose(infile3);
fclose(outfile);
}


void conv(float f[], float g[], float c[], int size)
{
  int m,k;

  for(k=0;k<size;k++)
   { c[k]=0;
     for(m=0;m<=k;m++)
      c[k] = c[k]+f[m]*g[k-m];
   }
}
```

```
/* Open the input and output data files. */

 outfile=fopen(argv[4],"w");

 for(i=0;i<xcsize;i++)
  { f1[i]=0;
    f2[i]=0;
  }
 for(i=0;i<fil_xcsize;i++)
  { xc[i]=0;
    b[i]=0;
    fil_xc[i]=0;
  }


 if((infile1=fopen(argv[1],"r")) == NULL)
  { printf("There is no data file %s!\n", argv[1]);
    exit(0);
  }
 else
  { i=0;
    while((fscanf(infile1,"%f",&f1[i])) !=EOF)
     i++;
  }


 if((infile2=fopen(argv[2],"r")) == NULL)
  { printf("There is no data file %s!\n", argv[2]);
    exit(0);
  }
 else
  { i=0;
    while((fscanf(infile2,"%f",&f2[i])) !=EOF)
     i++;
  }

 if((infile3=fopen(argv[3],"r")) == NULL)
  { printf("There is no data file %s!\n", argv[3]);
    exit(0);
  }
 else
  { i=0;
    while((fscanf(infile3,"%f",&b[i])) !=EOF)
       i++;
  }


 /* Calculate the cross-correlation of the input and the */
 /* reference signals.                                   */

  fliplr(f1,N);
  conv(f1,f2,xc,xcsize);

 /* To pass xc through a LPF */
  conv(b,xc,fil_xc,fil_xcsize);


  /* Find the shift, using direct max. correlation technique. */
  if(method==2)
```